μGPCsH シリーズ

命令語編　取扱説明書

はじめに

このたびは、TOYO FA ディジタルコントローラμGPCsH をお買い上げいただきまことにありがとうございます。

このプログラミングマニュアル　命令語編は、プログラミングの考え方、リレーおよびレジスタの説明、各命語について解説したものです。μGPCsH を正しくお使いいただくために、このプログラミングマニュアル　命令語編をよくお読みください。

また、下表に示す関連マニュアルも併せてお読みくださるようお願いいたします。

名称	マニュアル番号	記載内容
μGPCsH シリーズ　プログラミングマニュアル（オペレーション編）	QG18274	TDFlowEditor のメニュー、アイコンなどの説明および TDFlowEditor のオペレーションのすべてを解説
μGPCsH シリーズ　ユーザーズマニュアル（ハードウェア編）	QG18284	μGPCsH シリーズのシステム構成、各モジュールのハードウェア仕様などを解説

ご注意

（１）　本書の内容の一部または全部を無断で転載、複製することは禁止されております。

（２）　本書の内容に関しては、改良のため予告なしに仕様などを変更することがありますのでご了承ください。

（３）　本書の内容に関しては万全を期しておりますが、万一ご不審な点や誤りなどお気付きのことがありましたら、お手数ですが巻末記載の弊社営業所までご連絡ください。その際、表紙記載のマニュアル番号も併せてお知らせください。

安全上のご注意

本製品をご使用の前に「安全上のご注意」をよくお読みの上、正しくご使用ください。
ここでは、安全上の注意事項のレベルを「危険」および「注意」として区分しており、意味は下記のとおりです。

⚠危険：取り扱いを誤った場合に、死亡または重傷を受ける可能性があります。

⚠注意：取り扱いを誤った場合に、中程度の障害や軽傷を受ける可能性、あるいは物的損傷が発生する可能性があります。

なお、⚠ 注意に記載した事項でも、状況によっては重大な結果に結びつく可能性があります。

いずれも重要な内容を記載しておりますので、必ず守ってください。
特に注意していただきたい点を以下に示しますが、マニュアルの本文中にも上記記号で示します。

⚠危険
● 非常停止回路・インタロック回路などは、PC の外部で構成してください。 PCの故障により、機械の破損や事故のおそれがあります。

⚠注意
● 運転中のプログラム変更、強制出力、起動、停止などの操作は十分安全を確認してから行ってください。 操作ミスにより機械が動作し、機械の破損や事故のおそれがあります。

改定履歴

※マニュアル番号は、このマニュアルの表紙の右下に記載しております。

印刷日付	※マニュアル番号	改訂内容
2008 年 7 月	QG18273	初版発行
2009 年 8 月	QG18273	C_FREE 引数変更、M_RECV エラーステータス追記、M_SEND,MRECV サイズ表記修正
2010 年 10 月	QG18273	位相補償グラフ修正、現代制御追加

目　次

第1章 概要

μGPCsH シリーズでは、アプリケーションプログラム用の言語として、コンピュータ用言語(アセンブラ言語, C 言語など)を用いることなく、制御用言語としてμ-GPC 言語を開発しました。

μ-GPC 言語は、論理演算にはシーケンサなどで従来から使用されてきたラダーネットワークを、数値演算にはアナログコンピュータなどで使用されてきた D・F・S(データ・フロー・シンボル)を用いており、パーソナルコンピュータを利用したプログラミングツール上でビジュアルなプログラミングを可能とする新しいプログラム手法です。

μ-GPC 言語は次の特徴があります。

(1)コンピュータ言語の概念を変えた制御用として最適な言語体系です。
(マイクロプロセッサの処理手順を記述するのではなく、データの加工手順を記述します。)

(2)図式表示言語であり、プログラムが非常に分かり易く、誤りの少ないプログラミングが可能です。
(論理演算とデータ処理の両者を同一画面上でプログラミングすることが可能です。)

(3)取り扱うデータの種類(整数、BCD 型、実数等)を自動変換するため、プログラムのなかで型変換命令を使用する必要がありません。
(データを分割して使用する場合は変換命令が使用可能)

(4)S 字演算等の制御向け時系列関数が豊富に利用できるため、多数のラダーシンボルで実現した機能が 1 個のシンボルで記述され、誰でも簡単にプログラムすることができます。
(プログラムの実行時間を計測しながら自動調整していますので、時間意識は全く不要です。)

(5)3 つのインデックスレジスタ(X、Y、Z)によるインデックス修飾が可能であり、コンピュータ的な柔軟なプログラミングも可能です。
(ジャンプ命令を使用したプログラムループによるステップ数の減量にも効果あります。)

(6)サブプログラムを使った構造化プログラムを作成することが容易に可能です。
(アプリケーションプログラムの再利用・標準化に最適です。)

(7)4 個のマルチタスクブログラムを作成することが可能で、効率的なシステムが構築できます。(実行サイクルタイムを個別に設定可能ですので、実行周期を4分割することができます。)

(8)CPU 本体にプログラムに関するすべての情報をメモリしていますので、開発時に使用したパーソナルコンピュータが万一破損した場合でも別のパーソナルコンピュータで保守が可能です。
(プログラム上のコメントも再現されますので、プログラム・コメント・実行データのセットで保守可能)

(9)便利な機能を豊富に盛り込んだプログラミングツール(TDFlowEditor)を使用することによってシステム変更時の変更作業が極めて短時間に、誤りが少なく、確実に行うことができます
(RUN 中ローダ、モニター、デバッガ、トレンド、トレースバック機能等の詳細については TDFlowEditor オペレーションマニュアルを参照してください。)

第2章　μ－GPC言語でのプログラミング方法

μGPCsH では 1 台の CPU にロードされるプログラムをプロジェクトと言う概念で構築します。
プロジェクトには名称が付けられ、自由に変更することが可能です。（最適な名称を決めます。）
1 つのプロジェクトは IO 割付、タスク 1、タスク 2、タスク 3、タスク 4、サブルーチンの６つの部分に分けられます。

（1）IO割付

CPU のハード的な条件を定義するためのもので、システム構成を定義します。

（2）タスク 1、タスク 2、タスク 3、タスク 4

最も優先度の高いタスクをタスク 1 とし、スキャンタイム、複数のサブプログラムから構成されます。各サブプログラムにはプログラムの名称が付けられ（指定ない場合は NoName）、プロジェクト内での適当な担当処理名等に変更可能です。

1 つのサブプログラムは横 12 カラム、縦 19 ラインで構成されるプログラミングシートにプログラミングします。1 枚のプログラミングシートを 1 ページとし、順次ページを追加していくことが可能です。

サブプログラム内ではローカルシンボルが使用可能となりますが、サブプログラム間の受け渡しはグローバルメモリのみ有効です。

（3）サブルーチン

タスク 1、タスク 2、タスク 3、タスク 4 のなかのサブプログラムと同様に共通で使用されるサブルーチンです。

サブルーチンの名称（6 桁の英数字）を決め、追加します。

（4）プログラミングシート

横 12 カラムのうち、それぞれのカラムはシンボル挿入部分とクロスポイント部分から構成されます。これらの部分にシンボルを置き、ラベル名称を入力することでプログラミングが完了します。

（END 命令や、コンパイル操作はなく、エディタ終了時点で自動的にコンパイルされます。）

1～11 カラムはラダーシンボルでの接点およびデータフローシンボルが置けます。

12 カラムはラダーシンボルでのコイル専用でコイル以外は置けません。

また 11 カラムにはクロスポイントがありませんので加算命令やラダーシンボルの交点を挿入することはできません。

通常クロスポイントは 2 項演算子（加算、減算、乗算、等）を置きますが、C 接点のみは接点名称を入力するため、シンボル挿入部分に置きます。

縦 19 ラインのうち、それぞれの行（ライン）はラベル名称部分、シンボル挿入部分とデータ・コメント部分に分かれます。

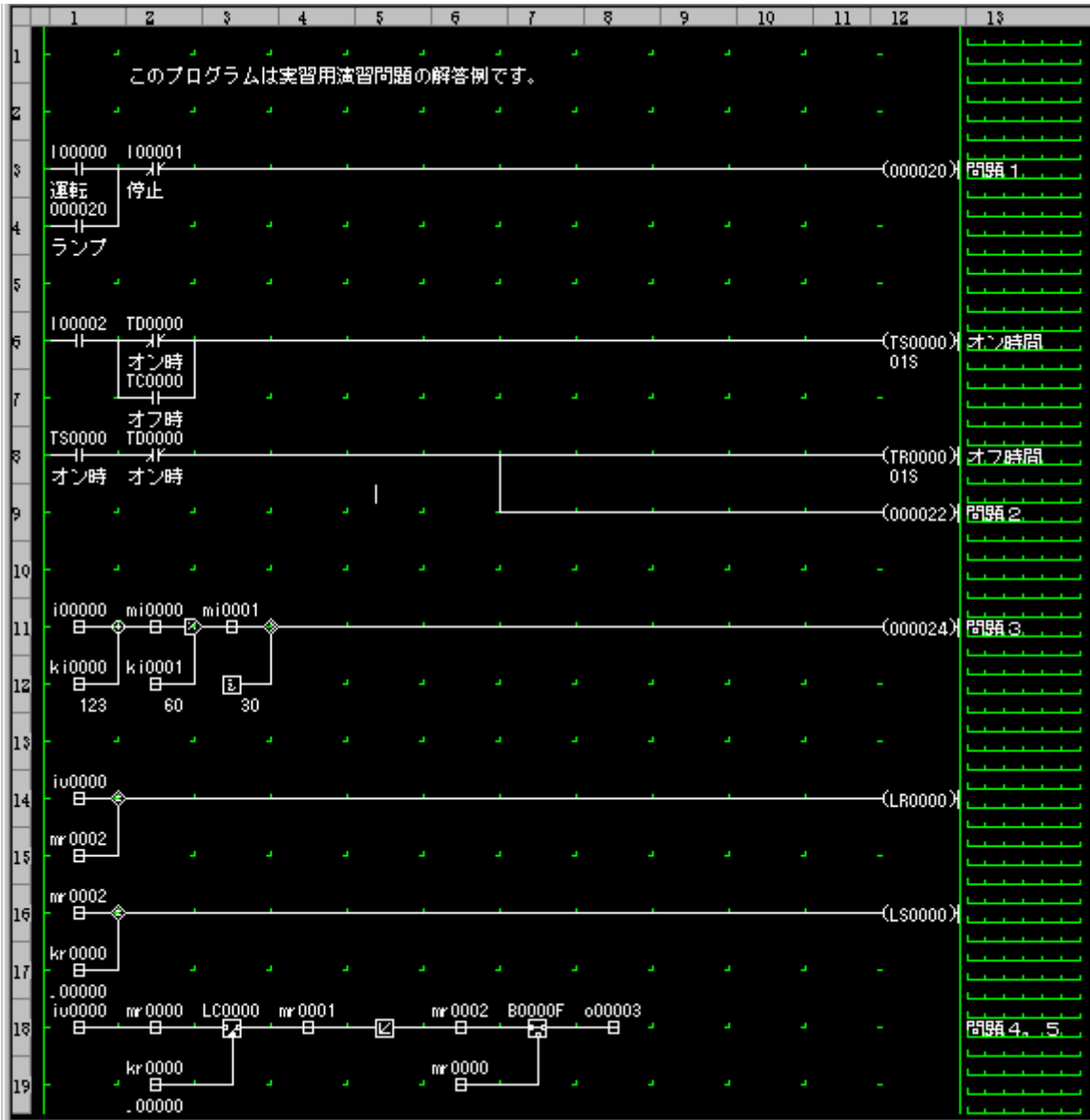
クロスポイントを使用したプログラムでは複数行に渡ってプログラムしますが、19 ラインを超えるプ

ログラムはテンポラリラベルを使用してページ分割します。

(5)プログラムコメント

プログラミングシートは下図プログラミング例のように 13 カラム目をコメント用に使用することができ、ラダーシンボルでコイルを置いた場合には該当接点部分のコメント位置に反映されます。(接点側で入力した場合以外、自動的に表示されます。)

ただし、全角で 3 桁(半角 6 桁)までですので、極力判別できる文字列を検討してください。

また、1 行目のようにシンボルがない部分のコメント位置はすべてコメントとして使用できます。

（6）サンプルプログラムの説明

参考までに前記実習用演習問題のプログラム例を説明します。

1 ライン目はコメント行です。この例のようにプログラムの内容等を予め記述します。

2 ライン目は空白行です。プログラムリストを見やすくするために必要に応じて入れておきます。

3 ライン目～4 ライン目は典型的な 2 操作スイッチを使用したホールド回路のラダーシンボルです。
I00000 である入力スイッチをオンすることによって O00020 であるランプ回路を点灯させ、ホールドします。
I00001 は上記ホールドを解く B 接点入力スイッチです。オンすると上記ランプは消灯します。

5 ライン目は空白行です。

6 ライン目～9 ライン目はオンディレイタイマとオフディレイタイマを組み合わせたランプの点滅回路です。オン時間とオフ時間をそれぞれ独立に変更することができます。
各タイマの設定時間は 12 カラム目のコイルの下側に時間設定します。前記の例では 1.0S（秒）となっていますが、2 時間までの設定が可能で、H で時を、M で分を、S で秒を表します。最小単位は 10mS ですが、0.01Sと記述します。

10 ライン目は空白行です。

11 ライン目～12 ライン目は 16 ビットの入力モジュールから数値データを読み取り定数 123 を加算しさらに 60 で除算した余りを求め、その値が 30 を超えた場合ランプを点灯させる回路です。
途中の演算結果はレジスタにセーブしてありますので、デバッグ時はこれを確認しながら結果をモニターすることができます。比較命令シンボルの右側は論理演算シンボルとなります。

13 ライン目は空白行です。

14 ライン目～19 ライン目はラッチリレーと変化率制限関数（弊社では ARC と称しています。）を使用したパターン発生回路の例です。連続的に三角波を発生します。波高値は入力モジュールから数値を BCD 型で設定することができます。周期は ARC 関数の変化率パラメータを変更することで間接的に変更することが可能です。18 ライン目と 19 ライン目で実数演算と整数演算、BCD 演算が混在していますが、ARC の入力値を C 接点で切り替えることによってパターンを連続的に発生させています。
B0000F の C 接点はテスト用で、デバッガでオンすることによって入力値を直接出力します。

第3章 取り扱えるデータの型と範囲

μGPCsH で取り扱うデータは種別 2 桁＋16 進番号 4 桁のラベル名称で表します。
また 16 進番号の先頭 1 桁はインデックスラベル X、Y、Z に置き換えできます。
ラベル例 ： I0X123　　b0y234　　mr02AF

3.1 データの種類

μGPCsH で取り扱うデータは大別すると、「論理データ」と「数値データ」の 2 種類があります。

３－１－１　論理データ
・論理データは、1 ビットの論理すなわち「1」または「0」を表すデータです。
・論理データは、論理演算などにより処理されます。
・論理データは、「リレー」に蓄えられており、プログラムの中では、「リレー番号」を指定することにより参照されます。
・比較演算シンボルの演算結果は論理データとなります。

要点 ・μGPCsH では論理データを蓄えておくところを「リレー」といいます。
・論理データの「1」はリレーの「ON」の状態に相当し、論理データの「0」はリレーの「OFF」の状態に相当します。

３－１－２　数値データ
・数値データは、16 ビット（1 ワード）または 32 ビット（2 ワード）を 1 単位として表すデータです。
・数値データは、「レジスタ」に蓄えられており、プログラムの中では、「レジスタ番号」を指定することにより参照されます。
・比較演算シンボルの入力条件は数値データとなります。

要点 ・μGPCsH では数値データを蓄えておくところを「レジスタ」といいます。

論理データのリレー番号の頭文字は大文字を使います。
（例）

I00000

数値データのレジスタ番号の頭文字は小文字を使います。
（例）

i00000

3.2　　データの型の種類

３－２－１　論理データの型

特に型の区別はありません。

取り扱えるデータは 1(ON)または 0(OFF)です。

３－２－２　数値データの型

下記の 5 種類があり、３－３以降に説明します。

① 16 ビット整数型(i 形式)
② 16 ビット BCD 型(u 形式)
③ 32 ビット整数型(w 形式)
④ 32 ビット BCD 型(v 形式)
⑤ 32 ビット実数型(r 形式)

3.3　　16 ビット整数型(i形式)

16 ビットの符号付き整数値データを 1 単位(1 ワード)として表します。

内部的に取り扱われるデータの範囲は

－32,768 ～ 32,767(8000H ～ 7FFFH)

このような数値データを「16 ビット整数データ」といいます。

3.4　　16 ビットBCD型(u形式)

16 ビットの BCD(2 進化 10 進コード)4 桁のデータを 1 単位(1 ワード)として表します。

内部的に取り扱われるデータの範囲は

0000 ～ 9999(0000H ～ 270FH)

このような数値データを「16 ビット BCD データ」といいます。

注意　16 ビット BCD データは、入出力ユニット(I／0)との間でやりとりするデータ(入出力データ)についてのみ使用可能です。

3.5　　32 ビット整数型（w形式）

32 ビットの符号付き整数値データを 1 単位（2 ワード占有）として表します。
内部的に取り扱われるデータの範囲は、

－2147483648 ～ 2147483647（80000000H ～ 7FFFFFFFH）

このような数値データを「32 ビット整数データ」といいます。

注意　32 ビット整数データは、入出力ユニット（I／0）との間でやりとりするデータ（入出力データ）についてのみ使用可能です。

3.6　　32 ビットBCD型（v形式）

32 ビットの BCD（2 進化 10 進コード）8 桁のデータを 1 単位（2 ワード占有）として表します。
内部的に取り扱われるデータの範囲は

00000000 ～ 99999999（00000000H ～ 05F5EOFFH）

このような数値データを「32 ビット BCD データ」といいます。

注意　32 ビット BCD データは、入出力ユニット（I／0）との間でやりとりするデータ（入出力データ）についてのみ使用可能です。

3.7　32ビット実数型（r形式）

32ビットの浮動小数点フォーマットのデータを1単位（2ワード占有）として表します。
内部的に取り扱われるデータの範囲は

$-6.2573187 \times 10^{38}$ ～ 6.2573187×10^{38}

このような数値データを「32ビット実数データ」といいます。

参考　32ビット実数データは、内部的に次のように取り扱われます。
（ユーザは気にする必要はありません。）

$(-1)^{S} \times 2^{e-127} \times 1.f$

s：符号部の値
e：指数部の値
f：仮数部の値（23ビット2進数で正規化）

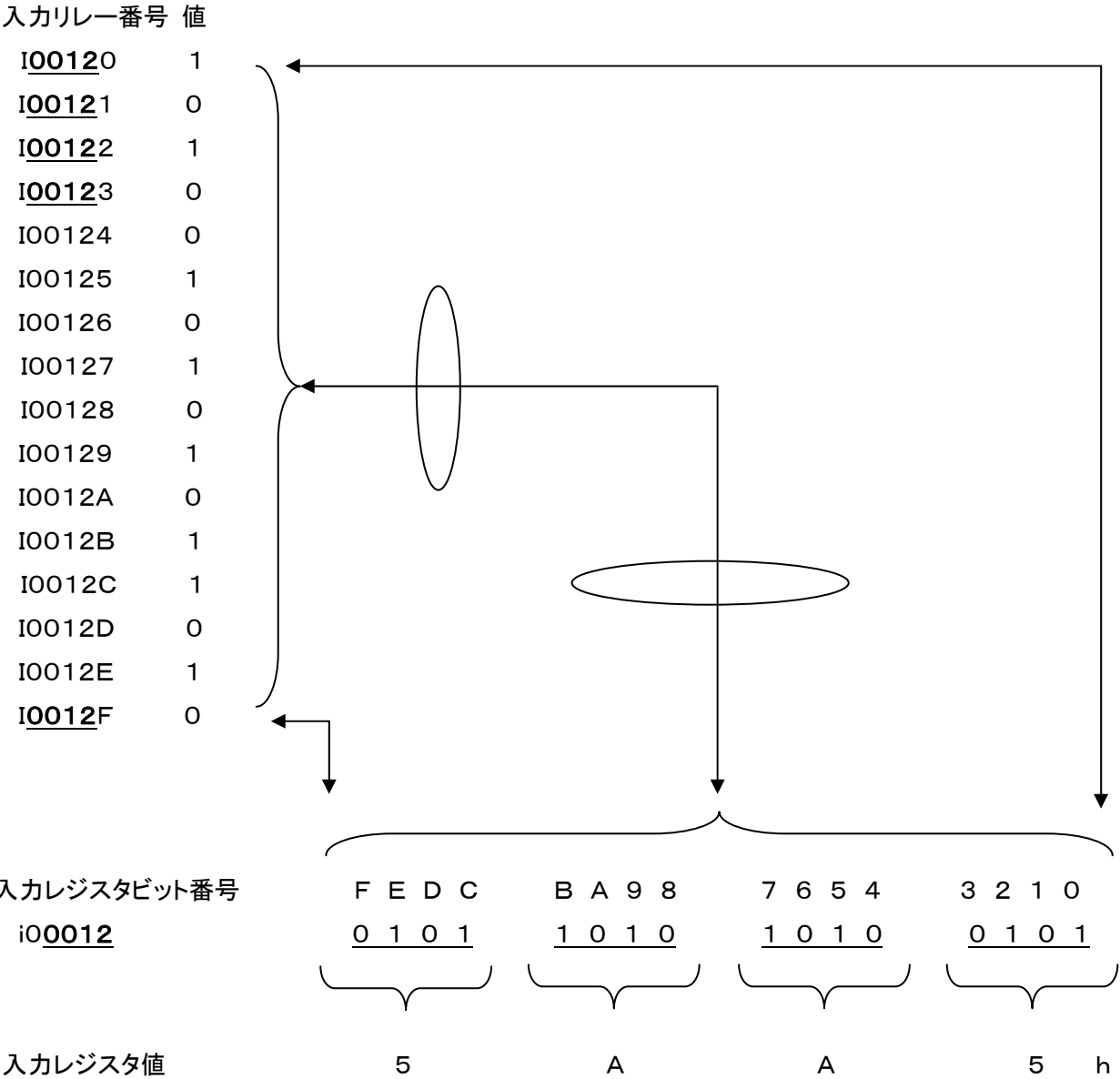
31	30 　23	22 　0
S	指数部	仮数部
1ビット	8ビット	23ビット

3.8 論理データと16ビット整数データ(i形式)との関係

μGPCsH で取り扱う「論理データ」は、16 ビットずつまとめて 1 つの「16 ビット整数データ(i 形式)」として対応づけることができます。

この場合の論理データと 16 ビット整数データ、およびこれらのデータを格納するリレーとレジスタ、リレー番号とレジスタ番号には次のような関係があります。

(例)連続するリレー番号 IOO120、IOO121、～IOO12F は、16 個の論理データを格納した入力リレーに対応します。一方レジスタ番号 iOOO12 は 1 個の 16 ビット整数データを格納する入力レジスタに対応します。両者の関係を図で示すと下図のようになります。

この図は入力レジスタ iOOO12 の内容 5AA5(16 進)が入力リレーIOO120、IOO121、～IOO12F に展開される様子を表したものです。

同様に 16 ビットずつまとめられる入力リレーと入力レジスタとの対応関係は次のようになります。

入力リレー番号			入力レジスタ番号
I00000, I00001,	～,	I0000F	i00000
I00010, I00011,	～,	I0001F	i00001
I00020, I00021,	～,	I0002F	i00002

この他、出力リレー、リンクリレー、補助リレー等それぞれリレーの種類別に、同様に出力レジスタ、リンクレジスタ、補助レジスタ等を対応づけることができます。

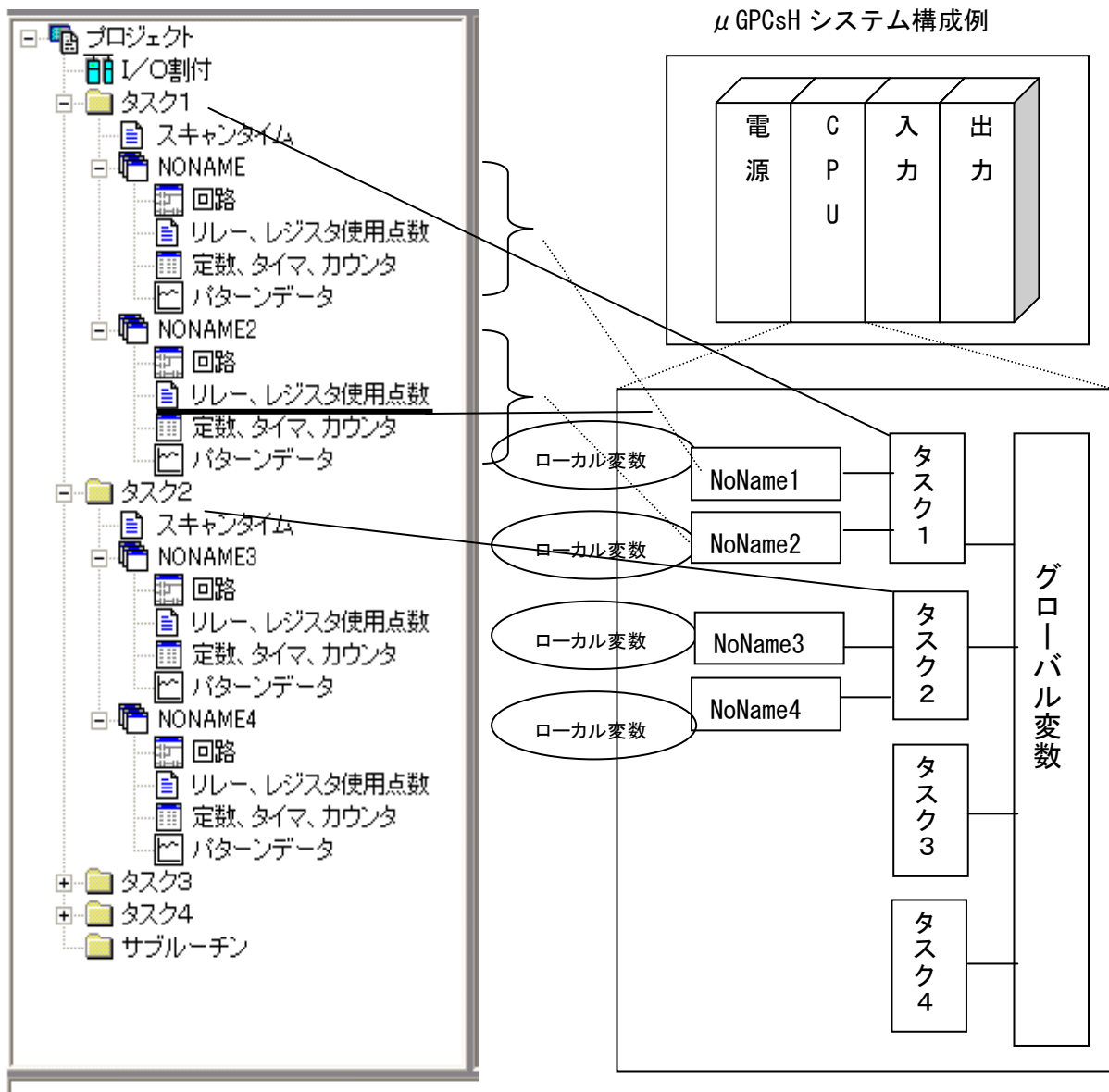
要点　リレー番号とレジスタ番号の対応関係

（例）

リレー番号 IOO123 はレジスタ番号 iOOO12 のビット番号 3 を表します。

注意 リレー番号とレジスタ番号の取り得る範囲はリレーやレジスタの種類によって異なります。
リレーに展開しても意味がなく、展開できないレジスタ（kr、mr、mi 等）もあります。

第4章 リレーとレジスタの種類

4.1 ローカル変数、グローバル変数とサブプログラムの関係

・ローカル変数 ------ 1 つのサブプログラム内でのみ参照可能な変数(他のサブプログラムからは参照できません)。

各サブプログラムの“リレー、レジスタ使用点数”で使用する点数を設定します。

処理機能により分割して作成します。

（例） mi、B0 など

・グローバル変数 --- 1 つのプロジェクト内のどのサブプログラムからも参照可能な変数。

（例） G0、fi、RI など

4.2　リレー・レジスタ使用可能点数

①グローバル変数

プロジェクト内のどのサブプログラムでも使用可能な変数の最大使用できる点数を下表にあらわします。

名称	点数 （最大）	種別	データ番号	データ方向	備考
入力リレー	8,192	接点	I00000 ～ I01FFF	ロード	*1 *3
入力レジスタ	512	入力データ	ix0000 ～ ix01FF		
出力リレー	(8,192)	コイル、接点	O00000 ～ O01FFF	ストア	*1 *3
出力レジスタ	(512)	出力データ	ox0000 ～ ox01FF		
アナウンスリレー アナウンスレジスタ	32,768	システム情報	Z00000 ～ Z07FFF	ロード	
	2,048		z00000 ～ z007FF		
グローバルリレー グローバルレジスタ	131,072	コイル、接点	G00000 ～ G1FFFF	ロード ストア	
	1,048,576	グローバル データ	g00000 ～ gFFFFF		
	32,768		gr0000 ～ grFFFE		*2
リテインリレー リテインレジスタ	65,536	コイル、接点	RI0000 ～ RIFFFF	ロード ストア	
	65,536	リテイン データ	ri0000 ～ riFFFF		
	32,768		rr0000 ～ rrFFFE		*2
ネットワークリレー ネットワークレジスタ	65,536	コイル、接点	FI0000 ～ FIFFFF	ロード ストア	
	4,096	ネットワーク データ	fi0000 ～ fi0FFF		
	2,048		fr0000 ～ fr0FFE		*2
	4,096	ネットワーク データ	ei0000 ～ ei0FFF	ロード ストア	
	2,048		er0000 ～ er0FFE		*2

*1：入力と出力の合計点数となります。

*2：奇数番号は使用できません。

*3：X 内は入出力レジスタの型式を表す u(BCD4 桁)、v(BCD8 桁)、w(32 ビット整数)があります。

②ローカル変数

各サブプログラム単位に最大使用できる点数を下表にあらわします。

名称	点数 （最大）	種別	データ番号	データ方向	備考
補助リレー	6144	コイル、接点	B00000 ～ B017FF	ロード ストア	
補助レジスタ	384	補助データ	b00000 ～ b0017F		
ラッチリレー ラッチレジスタ	512 32	セットコイル	LS0000 ～ LS01FF	ロード ストア	
			ls0000 ～ ls001F		
		リセットコイル	LR0000 ～ LR01FF	ロード ストア	
			lr0000 ～ lr001F		
		ラッチ接点	LC0000 ～ LC01FF	ロード	
			lc0000 ～ lc001F		
オン微分リレー オン微分レジスタ	512	コイル	US0000 ～ US01FF	ロード ストア	
			us0000 ～ us001F		
	32	微分接点	UC0000 ～ UC01FF	ロード	
			uc0000 ～ uc001F		
オフ微分リレー オフ微分レジスタ	512	コイル	DS0000 ～ DS01FF	ロード ストア	
			ds0000 ～ ds001F		
	32	微分接点	DC0000 ～ DC01FF	ロード	
			dc0000 ～ dc001F		
オンタイマ オンタイマ レジスタ	512	コイル、 瞬時接点	TS0000 ～ TS01FF	ロード ストア	
			ts0000 ～ ts001F		
	32	限時接点	TD0000 ～ TD01FF	ロード	
			td0000 ～ td001F		
	512	経過時間	tn0000 ～ tn01FF	ロード	
オフタイマ オフタイマ レジスタ	512	コイル、 瞬時接点	TR0000 ～ TR01FF	ロード ストア	
			tr0000 ～ tr001F		
	32	限時接点	TC0000 ～ TC01FF	ロード	
			tc0000 ～ tc001F		
	512	経過時間	tf0000 ～ tf01FF	ロード	

<table>
<tr><th>名称</th><th>点数
（最大）</th><th>種別</th><th>データ番号</th><th>データ方向</th><th>備考</th></tr>
<tr><td rowspan="11">カウンタ

カウンタレジスタ</td><td rowspan="10">256

16</td><td rowspan="2">リセット
コイル</td><td>NR0000 ～ NR00FF</td><td rowspan="2">ロード
ストア</td><td rowspan="2"></td></tr>
<tr><td>nr0000 ～ nr000F</td></tr>
<tr><td rowspan="2">プリセット
コイル</td><td>NP0000 ～ NP00FF</td><td rowspan="2">ロード
ストア</td><td rowspan="2"></td></tr>
<tr><td>np0000 ～ np000F</td></tr>
<tr><td rowspan="2">UP コイル</td><td>NU0000 ～ NU00FF</td><td rowspan="2">ロード
ストア</td><td rowspan="2"></td></tr>
<tr><td>nu0000 ～ nu000F</td></tr>
<tr><td rowspan="2">DOWN コイル</td><td>ND0000 ～ ND00FF</td><td rowspan="2">ロード
ストア</td><td rowspan="2"></td></tr>
<tr><td>nd0000 ～ nd000F</td></tr>
<tr><td rowspan="2">ゼロ検出接点</td><td>NZ0000 ～ NZ00FF</td><td rowspan="2">ロード</td><td rowspan="2"></td></tr>
<tr><td>nz0000 ～ nz000F</td></tr>
<tr><td>256</td><td>カウント現在値</td><td>N00000 ～ n000FF</td><td>ロード</td><td></td></tr>
<tr><td rowspan="2">演算データ</td><td>8192</td><td>整数</td><td>mi0000 ～ mi1FFF</td><td rowspan="2">ロード
ストア</td><td></td></tr>
<tr><td>4096</td><td>実数</td><td>mr0000 ～ mr0FFF</td><td></td></tr>
<tr><td rowspan="2">定数データ</td><td>8192</td><td>整数</td><td>ki0000 ～ ki1FFF</td><td rowspan="2">ロード</td><td></td></tr>
<tr><td>4096</td><td>実数</td><td>kr0000 ～ kr0FFF</td><td></td></tr>
<tr><td rowspan="2">パターンデータ</td><td>10</td><td>整数</td><td>pi0000 ～ pi0009</td><td rowspan="2">ロード</td><td>*1</td></tr>
<tr><td>10</td><td>実数</td><td>pr0000 ～ pr0009</td><td>*1</td></tr>
<tr><td rowspan="3">スタックレジスタ</td><td>4096</td><td>コイル、接点</td><td>SI0000 ～ SIFFFF</td><td rowspan="3">ロード
ストア</td><td rowspan="2"></td></tr>
<tr><td>256</td><td>整数</td><td>si0000 ～ si00FF</td></tr>
<tr><td>128</td><td>実数</td><td>sr0000 ～ sr00FF</td><td>*2</td></tr>
<tr><td>インデックスレジスタ</td><td>3</td><td>整数</td><td>indx_x、indx_y、
indx_z</td><td>ロード
ストア</td><td></td></tr>
</table>

*1：パターンデータのポイント数の設定によって使用可能なパターン数も変わります。

*2：奇数番号は使用できません。

③レジスタの共有体構造

グローバルレジスタやスタックレジスタは取り扱いを良くするために共有体関係になっています。

下表にグローバルメモリのリレー、整数レジスタ、実数レジスタの共有体関係を表します。

特に sr0000 は活線データを、sr0002 は引数の 1 番目をあらわします。

<table>
<tr><th>リレー名称</th><th>整数レジスタ</th><th>実数レジスタ</th></tr>
<tr><td>G00000</td><td rowspan="3">g00000</td><td rowspan="6">gr0000</td></tr>
<tr><td>G00001</td></tr>
<tr><td>G00002
G0000F</td></tr>
<tr><td>G00010</td><td rowspan="3">g00001</td></tr>
<tr><td>G00011</td></tr>
<tr><td>G00012
G0001F</td></tr>
<tr><td>G00020
G0002F</td><td>g00002</td><td rowspan="2">gr0002</td></tr>
<tr><td>G00030
G0003F</td><td>g00003</td></tr>
</table>

<table>
<tr><th>リレー名称</th><th>整数レジスタ</th><th>実数レジスタ</th></tr>
<tr><td>SI0000</td><td rowspan="3">si0000</td><td rowspan="6">sr0000</td></tr>
<tr><td>SI0001</td></tr>
<tr><td>SI0002
SI000F</td></tr>
<tr><td>SI0010</td><td rowspan="3">si0001</td></tr>
<tr><td>SI0011</td></tr>
<tr><td>SI0012
SI001F</td></tr>
<tr><td>SI0020
SI002F</td><td>si0002</td><td rowspan="2">sr0002</td></tr>
<tr><td>SI0030
SI003F</td><td>si0003</td></tr>
</table>

注意:共有体関係はいずれのレジスタからも操作できますので使用の場合は特に注意してください。

④CPUアナウンスレジスタ

レジスタ名	リレー名	名称	内容
z00000	Z00000	CPU　RUN	CPUが運転中でONするリレー
	Z00001	重故障	CPUが重故障でONするリレー
	Z00002	軽故障	CPUが軽故障でONするリレー
z00003	—	スキャンタイム1	タスク1　スキャンタイムレジスタ(BCD)msec
z00004	—	スキャンタイム2	タスク2　スキャンタイムレジスタ(BCD)msec
z00005	—	時計レジスタ(年月)	年(H側)、月(L側)の表示(BCD)
z00006	—	時計レジスタ(日時)	日(H側)、時(L側)の表示(BCD)
z00007	—	時計レジスタ(分秒)	分(H側)、秒(L側)の表示(BCD)
z00008	—	未使用	常時0
z00009	—	0．25msカウンタ	0．25ms毎に加算されるカウンタ
z0000A	—	1secカウンタ	1秒毎に加算されるカウンタ
z0000B	—	システムタスクカウンタ	システムタスク起動毎に加算されるカウンタ
z0000C	—	局番スイッチ情報	FL－net局番スイッチの値(00h～FFh「255」)
z0000D	Z000D0	CPU実装情報	CPUスロット実装情報(常時0)
	Z000D1	IO1実装情報	IO1スロット実装情報(実装あり:0　実装なし:1)
	Z000D2	IO2実装情報	IO2スロット実装情報(実装あり:0　実装なし:1)
	Z000D3	IO3実装情報	IO3スロット実装情報(実装あり:0　実装なし:1)
	Z000D4	IO4実装情報	IO4スロット実装情報(実装あり:0　実装なし:1)
	Z000D5	IO5実装情報	IO5スロット実装情報(実装あり:0　実装なし:1)
	Z000D6	IO6実装情報	IO6スロット実装情報(実装あり:0　実装なし:1)
	Z000D7	IO7実装情報	IO7スロット実装情報(実装あり:0　実装なし:1)
	Z000D8	IO8実装情報	IO8スロット実装情報(実装あり:0　実装なし:1)
	Z000D9	IO9実装情報	IO9スロット実装情報(実装あり:0　実装なし:1)
	Z000DA	未使用	常時1
	Z000DB	未使用	常時1
	Z000DC	USB接続	TOOL　I／F　USB接続:0　USB未接続:1
	Z000DD	CPU実装	CPU実装情報(常時1)
	Z000DE	電池電圧	電池電圧正常または電池なし:1　電池電圧低下:0
	Z000DF	RUN／STOPレバー	RUN:1　STOP:0
z0000E	Z000E0 ～ Z000E7	未使用	常時0
	Z000E8	操作スイッチ　ENT	ENTボタン押:1　ENTボタン離:0
	Z000E9	操作スイッチ　D	DUレバー　D側:1　中立またはU側:0
	Z000EA	操作スイッチ　U	DUレバー　U側:1　中立またはD側:0
	Z000EB	操作スイッチ　L	LRレバー　L側:1　中立またはR側:0
	Z000EC	操作スイッチ　R	LRレバー　R側:1　中立またはL側:0
	Z000ED ～ Z000EF	未使用	常時0
z0000F	—	CPUバージョン	CPUバージョンレジスタ　1．00　＝　100

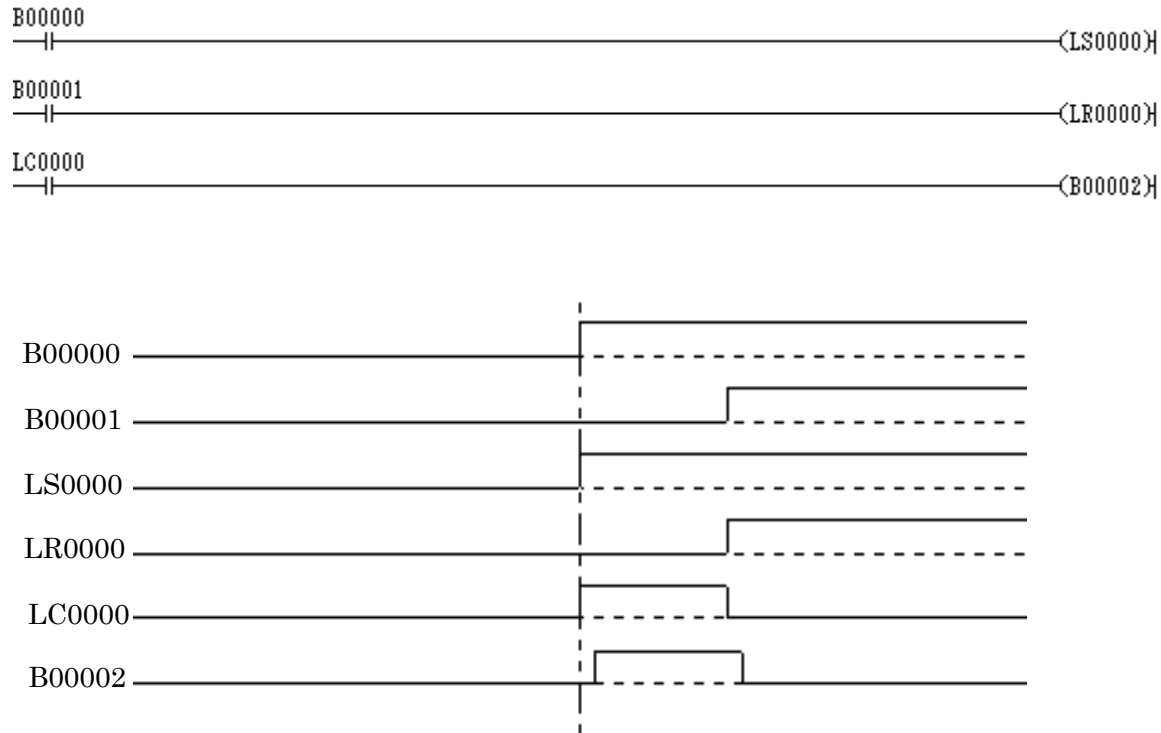
④CPUアナウンスレジスタ つづき1

レジスタ名	リレー名	名称	内容
z00010 ～ z00017	—	IO初期化エラー	IO初期化エラー又はTr出力モジュールヒューズ切れ Z001X0～Z001XF:スロット番号 Z0010X～Z0017X:ユニット番号(基本ユニット:0)
z00018 ～ z0001F	—	IOオンラインチェックエラー	IOオンラインチェックエラー又はTr出力モジュール外部電源切れ Z001X0～Z001XF:スロット番号 Z0018X～Z001FX:ユニット番号(基本ユニット:0)
z00020 ～ z00027	—	IO構成変化	IOモジュールの情報が変化した Z002X0～Z002XF:スロット番号 Z0020X～Z0027X:ユニット番号(基本ユニット:0)
z00030 ～ z00037	—	IO設定異常	IO割り付けと実構成が違う Z003X0～Z003XF:スロット番号 Z0030X～Z0037X:ユニット番号(基本ユニット:0)
z00038 ～ z0004F	—	未使用	未使用
z00050 ～ z000FF	—	未使用	未使用(過去アプリケーション移植時の関数用コントロールリレーとして使用可能)
z00100 ～ z0012F	—	未使用	未使用
z00130	—	ローカルメモリ使用数	ローカルメモリ使用ワード(変数部:b0、mi、mr・・・) 最大131072ワード(L側のみ表示)
z00131	—	ローカルメモリ使用数	ローカルメモリ使用ワード(パラメータ部:ki、kr・・・) 最大65536ワード
z00132	—	コード使用数(L)	コード使用ワード(L) 最大327680ワード
z00133	—	コード使用数(H)	コード使用ワード(H) 最大327680ワード
z00134	—	システム定義使用数	システム定義使用ワード 最大8192ワード
z00135	—	未使用	未使用
z00136	—	未使用	未使用
z00137	—	汎用ファイル使用数	汎用ファイル情報使用ワード 最大131072ワード
z00138	—	IPアドレス	自モジュールIPアドレス(LL)
z00139	—	IPアドレス	自モジュールIPアドレス(LH)
z0013A	—	IPアドレス	自モジュールIPアドレス(HL)
z0013B	—	IPアドレス	自モジュールIPアドレス(HH)
z0013C ～ z0013F	—	未使用	未使用
z00140	—	自己診断用	自己診断用レジスタ(使用禁止)
z00141 ～ z0014F	—	未使用	未使用

④CPUアナウンスレジスタ つづき2

レジスタ名	リレー名	名称	内容
z00150	—	実行時間レジスタ	IOリフレッシュ実行時間(単位ms)(BCD)
z00151	—	スキャンタイムレジスタ	IOリフレッシュ起動周期(単位ms)(BCD)
z00152	—	実行時間レジスタ	タスク1実行時間(単位ms)(BCD)
z00153	—	スキャンタイムレジスタ	タスク1起動時間(単位ms)(BCD)
z00154	—	実行時間レジスタ	タスク2実行時間(単位ms)(BCD)
z00155	—	スキャンタイムレジスタ	タスク2起動時間(単位ms)(BCD)
z00156	—	実行時間レジスタ	タスク3実行時間(単位ms)(BCD)
z00157	—	スキャンタイムレジスタ	タスク3起動時間(単位ms)(BCD)
z00158	—	実行時間レジスタ	タスク4実行時間(単位ms)(BCD)
z00159	—	スキャンタイムレジスタ	タスク4起動時間(単位ms)(BCD)
z0015A	—	優先度レジスタ	IOリフレッシュ RTOS内タスク優先度
z0015B	—	優先度レジスタ	タスク1 RTOS内タスク優先度
z0015C	—	優先度レジスタ	タスク2 RTOS内タスク優先度
z0015D	—	優先度レジスタ	タスク3 RTOS内タスク優先度
z0015E	—	優先度レジスタ	タスク4 RTOS内タスク優先度
z0015F	—	バンクレジスタ	現在使用プログラムバンクレジスタ 1 or 2
zr0160	—	スキャンタイムレジスタ	IOリフレッシュ起動行時間(実数:単位秒)
zr0162	—	スキャンタイムレジスタ	タスク1起動時間(実数:単位秒)
zr0164	—	スキャンタイムレジスタ	タスク2起動時間(実数:単位秒)
zr0166	—	スキャンタイムレジスタ	タスク3起動時間(実数:単位秒)
zr0168	—	スキャンタイムレジスタ	タスク4起動時間(実数:単位秒)
zr016A ~ zr016E	—	未使用	未使用
zr016F		プログラム切替レジスタ	プログラム切替中:1
zr0170	—	実行時間レジスタ	IOリフレッシュ実行時間(実数:単位秒)
zr0172	—	実行時間レジスタ	タスク1実行時間(実数:単位秒)
zr0174	—	実行時間レジスタ	タスク2実行時間(実数:単位秒)
zr0176	—	実行時間レジスタ	タスク3実行時間(実数:単位秒)
zr0178	—	実行時間レジスタ	タスク4実行時間(実数:単位秒)
zr017A ~ zr017F	—	未使用	未使用
z00180	—	IO ｴﾗｰ発生箇所	″00US″ システム構成定義異常(定義なし実装あり)
z00181	—	IO ｴﾗｰ発生箇所	″00US″ システム構成定義異常(定義あり実装なし)
z00182	—	IO ｴﾗｰ発生箇所	″00US″ I/O モジュール異常 (IO ID Er)
z00183	—	IO ｴﾗｰ発生箇所	″00US″ I/O モジュール異常 (IODef Er)
z00184	—	IO ｴﾗｰ発生箇所	″00US″ 共通モジュール異常(IOFaltEr)
z00185	—	IO ｴﾗｰ発生箇所	″00US″ メモリバスアクセス異常(BusAccEr)
z00186	—	ｼｽﾃﾑｶｳﾝﾄﾚｼﾞｽﾀ	FInet データ転送タスク起動回数
z00187	—	ｼｽﾃﾑｶｳﾝﾄﾚｼﾞｽﾀ	FInet データ転送タスク起動周期(μs)
z00188	—	ｼｽﾃﾑｶｳﾝﾄﾚｼﾞｽﾀ	NULL タスク起動回数
z00189	—	ｼｽﾃﾑｶｳﾝﾄﾚｼﾞｽﾀ	NULL タスク起動周期(μs)
z0018A	—	ｼｽﾃﾑｶｳﾝﾄﾚｼﾞｽﾀ	IO リフレッシュ OS 周期

4.3　　特殊リレーの概要

①ラッチリレー／レジスタ

セットコル LS0000 が ON すると、ラッチ接点 LC0000 が ON し、O00020 は ON し続けます。

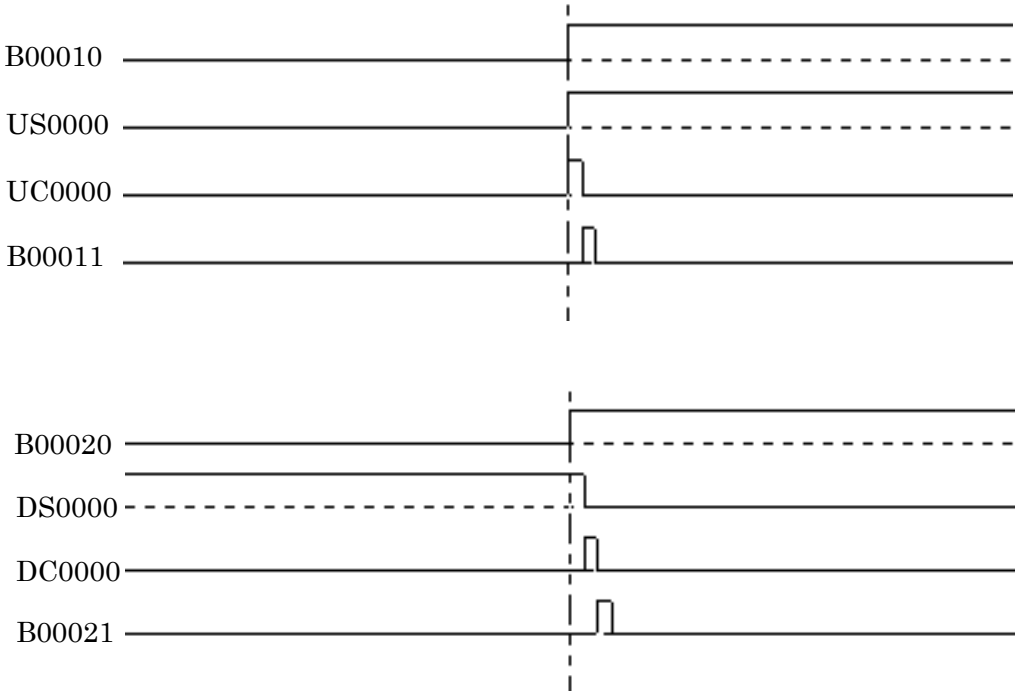
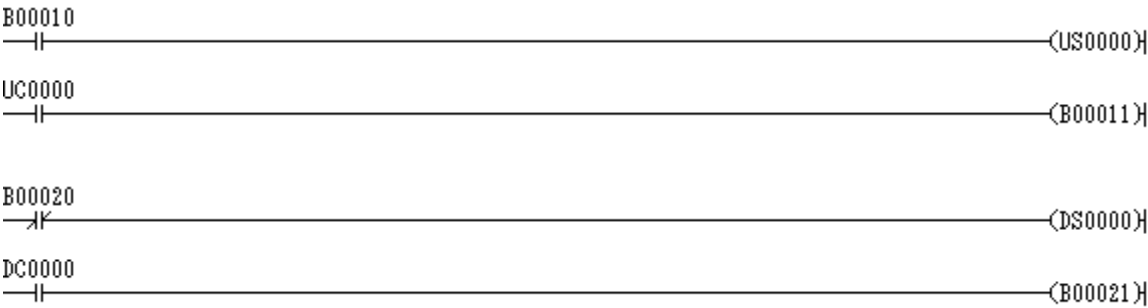
リセットコイル LR0000 が ON すると、ラッチ接点 LC0000 が OFF し、O00020 は OFF し続けます。

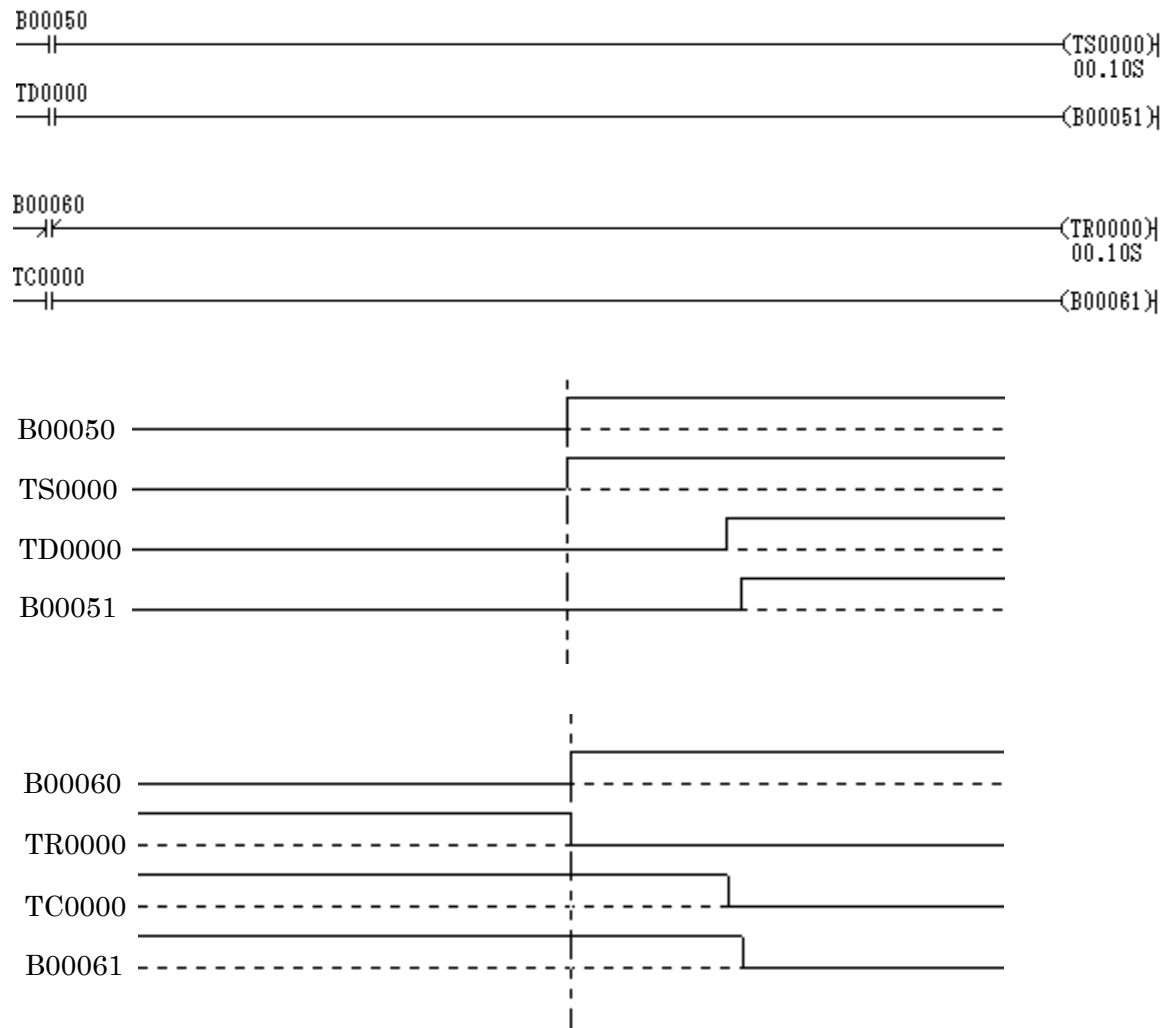
ラッチ接点 LC0000 はラッチコイルより1スキャン分遅れます。

ラッチコイルは通常電源を開放すると、OFF します。

ラッチコイルを電源開放時にも保持したい場合は、リテインメモリを使用しメモリ転送定義で転送するか、SET　RESET 関数を使用して（パラメータにリテインリレーを設定）ください。

サブルーチン内で同様の機能を実現するにはサブルーチン内で SI0000 を使用して SET　RESET 関数を使います。

②オン／オフ微分リレー／レジスタ

コイル US0000 が ON すると、1 スキャン分遅れて微分接点 UC0000 が 1 スキャン分 ON になります。
コイル DS0000 が OFF すると 1 スキャン分遅れて微分接点 DC0000 が1スキャン分 ON になります。
この他に同様の機能を実現するために USUC 関数と DSDC 関数があります。

③オン／オフタイマリレー／レジスタ

コイル TS0000 が ON すると、設定時間経過後に限時接点 TD0000 が ON になります。TD0000 は TS0000 が OFF になって 1 スキャン以内に OFF になります。

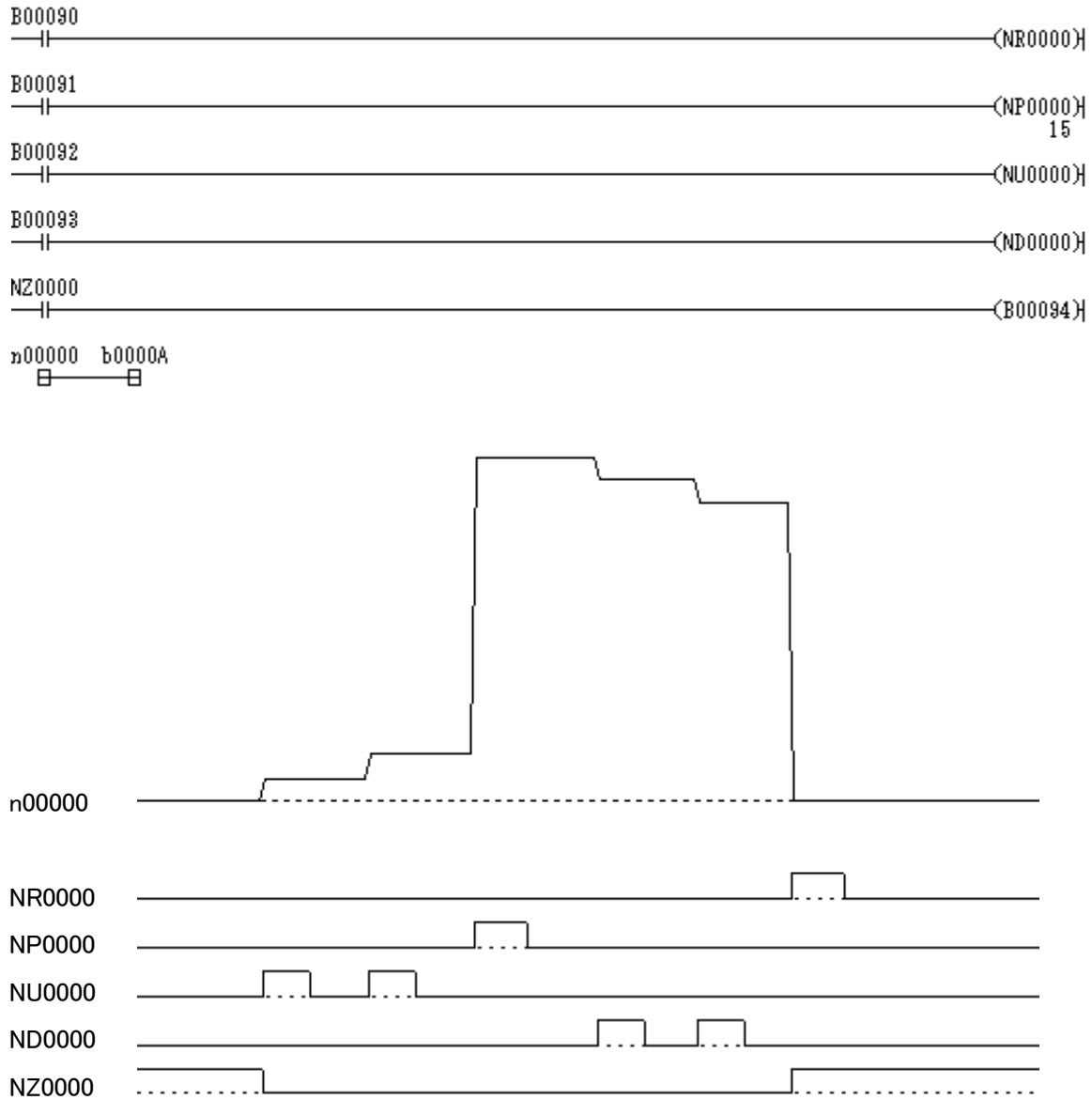
（タイマ設定値は TS コイルの下側に入力します。）

ここで S は秒を、M は分を、H は時間を意味し、0.01 秒から 2 時間までの設定が可能です。

コイル TR0000 が ON すると、限時接点 TC0000 は TR0000 が ON になってから 1 スキャン以内に ON になります。そして設定時間経過後 OFF になります。

（タイマ設定値は TR コイルの下側に入力します。）

ここで S は秒を、M は分を、H は時間を意味し、0.01 秒から 2 時間までの設定が可能です。

④カウンタリレー／レジスタ

カウンタの初期値は 0 です。次にアップコイルが ON となり、カウント値は 1 加えられます。またゼロ検出接点は始め 0 で ON ですが 1 が加えられたので、0 ではなく OFF となります。

さらにアップコイルが ON となり、カウント値は 1 加えられ 2 となります。

プリセットコイルが ON となりカウント値は 15 となります。

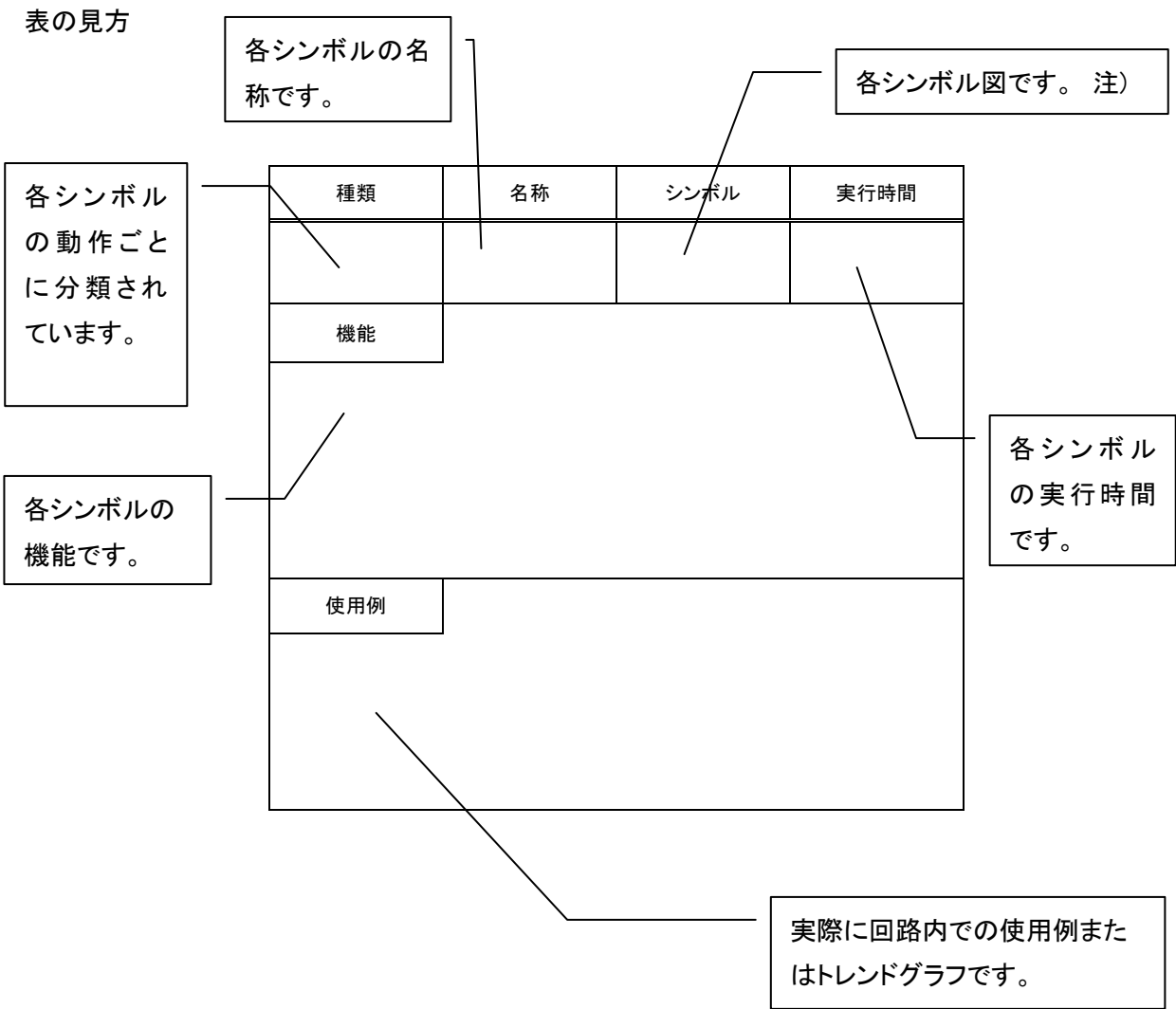
プリセット値は NP コイルの下側に設定します。

ダウンコイルが ON となりカウント値から 1 減らします。

さらにダウンコイルが ON となりカウント値から 1 減らします。

リセットコイルが ON となりカウント値は 0 となり、ゼロ検出接点は ON となります。

第5章 命令語説明

表の見方

注) ここではこれ以降のシンボル欄に表示される RELAY、REG について説明します。

RELAY

左図はリレーを示します。ここでは簡略化するために RELAY という言葉を用いて表します。RELAY には、G0、I0、B0 などのすべてのリレーが設定可能です。

REG

左図はレジスタを示します。ここでは簡略化するために REG という言葉を用いて表します。REG には、g0、mi、kr などのすべてのレジスタが設定可能です。

種類	名称	シンボル	実行時間
LD 言語	A 接点	RELAY —||—	0.10 ［μs］
機能	RELAY が ON ならば入力論理値を出力します。 OFF ならば出力論理値を OFF にします。		

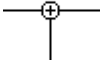
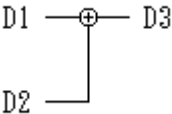
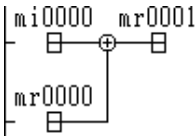
RELAY
A —||— B

RELAY	A	B
ON	ON	ON
ON	OFF	OFF
FF	X	OFF

X ： don't care

使用例

```
|B00000  B00001
|--||------||-------------------------------------------(B00010)|
```

リレーB00000 とリレーB00001 がともに ON の時は、リレーB00010 が ON になります。
これ以外では、リレーB00010 が OFF になります。

種類	名称	シンボル	実行時間
LD 言語	B 接点	RELAY ─┤/├─	0.12 ［μs］
機能	RELAY が OFF ならば入力論理値を出力します。 ON ならば出力論理値を OFF にします。		

RELAY
A ─┤/├─ B

RELAY	A	B
OFF	O	ON
OFF	OFF	OFF
ON	X	OFF

X ： don't care

使用例

B00000 B00001 (B00010)

リレーB00000 が ON、リレーB00001 が OFF の時はリレーB00010 が ON になります。
これ以外ではリレーB00010 が OFF になります。

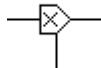
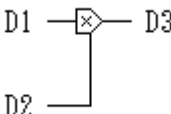
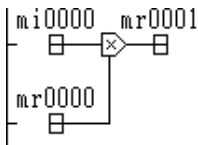
種類	名称	シンボル	実行時間
LD 言語	論理反転	─[~]─	0.10 [μs]
機能	入力値を論理反転します。		

A [~]─ B

A	B
ON	OFF
OFF	ON

使用例

B00000 ─┤├─[~]─ (B00001)

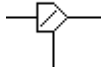
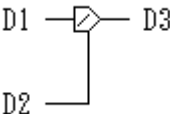
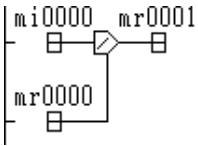
リレーB00000 が ON の時は、リレーB00001 が OFF になります。
リレーB00000 が OFF の時は、リレーB00001 が ON になります。

種類	名称	シンボル	実行時間
LD 言語	コイル	─(RELAY)	0.22 ［μs］
機能	入力論理値を RELAY に出力します。		

A ─(RELAY)

A	RELAY
ON	ON
OFF	OFF

使用例

```
I00000
─┤├──────────────────────────(O00020)
O00020
─┤├──────────────────────────(B00000)
```

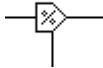
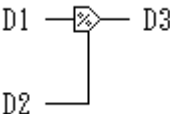
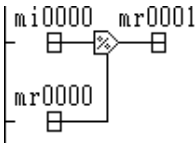
リレーI00000 が ON の時は、リレーO00020 とリレーB00000 がともに ON になります。
リレーI00000 が OFF の時は、リレーO00020 とリレーB00000 がともに OFF になります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	ロード	REG ⊟—	整数 0.16 [μs] 実数 0.20 [μs]
	ストア	REG —⊟	
機能	ロード：REG のデータを出力数値にします。 ストア：入力数値を REG に出力します。		

```
REG
 ⊟— D1            D1  =  REG


      REG
D2  —⊟            REG  =  D2
```

使用例

```
|ki0000   mi0000
├ ⊟――――――――⊟
       2
|mi0000   mr0000
├ ⊟――――――――⊟
```

レジスタ ki0000 のデータ(2)がロードされ、レジスタ mi0000 にストアされます。
次にレジスタ mi0000 のデータがロードされ、レジスタ mr0000 にストアされます。
レジスタ mr0000 は実数型のレジスタですので、整数・実数の型変換が行われデータ(2.0)がストアされす。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	ストア＆ロード ストア	REG ─□─	整数 0.19 ［μs］ 実数 0.14 ［μs］
機能	入力数値を REG に出力し、REG のデータを出力の数値にします。 演算の途中のデータを REG に保持する時に使用します。		

```
     REG
D1 ──□── D2
```

REG = D1

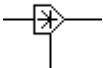
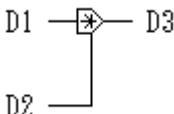
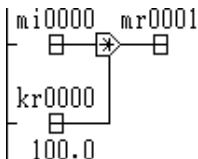
D2 = REG

使用例

```
|mi0000  mi0002  mi0004
├ □──⊕──□──⊖──□
|       |       |
|mi0001 |mi0003 |
├ □─────┘ □─────┘
```

レジスタmi0000のデータとレジスタmi00001のデータが加算され、結果がレジスタmi0002にストアされます。

次にレジスタ mi0002 のデータからレジスタ mi0003 のデータが減算され、結果がレジスタ mi0004 にストアされます。

レジスタ mi0002 には演算の途中の加算データが保持されることになります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	加算		整数 0.24 ［μs］ 実数 0.15 ［μs］
機能	2つの入力数値を加算し結果を出力します。 型が違っても演算できます。ただし、整数は実数に変換後、実数演算します。		

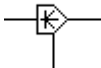
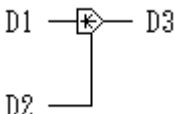
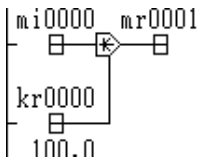
型変換について

1つの演算ブロックで使用しているレジスタの型式が整数型と16ビットBCD型の場合は16ビット整数型に変換して演算しますが、実数型、32ビット整数型、32ビットBCD型のレジスタを使用している場合は実数型に変換して演算します。

（以後、減算、乗算、除算、剰余、上位優先、下位優先についても型変換を行います。）

使用例

```
mi0000   mr0001
 □──⊕──□
     │
mr0000   │
 □───┘
```

レジスタmi0000のデータとレジスタmr0000のデータが加算され、結果がレジスタmr0001にストアされます。
レジスタmi0000のデータは整数ですが、レジスタmr0000のデータが実数ですので整数/実数の型変換を行ってから加算されます。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	減算		整数 0.28 ［μs］ 実数 0.18 ［μs］

機能	2 つの入力数値を減算し結果を出力します。 型が違っても演算できます。ただし、整数は実数に変換後、実数演算します。

D1 D3

D2

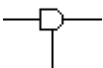
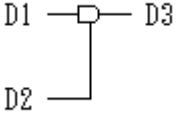
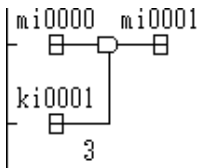
D3 = D1 - D2

使用例

mi0000 mr0001

mr0000

レジスタ mi0000 のデータからレジスタ mr0000 のデータが減算され、結果がレジスタ mr0001 にストアされます。

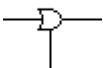
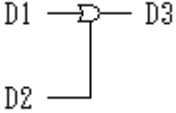
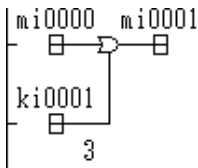
レジスタ mi0000 のデータは整数ですが、レジスタ mr0000 のデータが実数ですので整数/実数の型変換を行ってから減算されます。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	乗算		整数 0.26 ［μs］ 実数 0.23 ［μs］
機能	2 つの入力数値を乗算し結果を出力します。 型が違っても演算できます。ただし、整数は実数に変換後、実数演算します。		

D1 D3
D2

D3 = D1 ＊ D2

使用例

mi0000 mr0001
mr0000

レジスタmi0000のデータとレジスタmr0000のデータが乗算され、結果がレジスタmr0001にストアされます。
レジスタ mi0000 のデータは整数ですが、レジスタ mr0000 のデータが実数ですので整数/実数の型変換を行ってから乗算されます。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	除算		整数 0.69 ［μs］ 実数 0.38 ［μs］

機能	2 つの入力数値を除算し結果を出力します。 型が違っても演算できます。ただし、整数は実数に変換後、実数演算します。

D1　D3

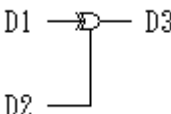
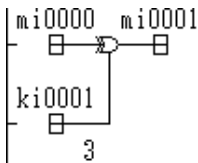
D2

D3 ＝ D1 ／ D2

使用例

mi0000　mr0001

mr0000

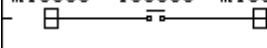
レジスタ mi0000 のデータとレジスタ mr0000 のデータが除算され、結果がレジスタ mr0001 にストアされます。
レジスタ mi0000 のデータは整数ですが、レジスタ mr0000 のデータが実数ですので整数/実数の型変換を行ってから除算されます。

種類	名称	シンボル	実行時間
データフロー言語 (基本)	剰余		0.64 ［μs］
機能	2 つの入力数値を除算し結果(剰余)を出力します。		

注） 整数演算のみ有効です。

使用例

mi0000 mr0001

mr0000

レジスタ mi0000 のデータがレジスタ mi0001 のデータで除算され、結果(剰余)がレジスタ mi0002 にストアされます。

種類	名称	シンボル	実行時間
データフロー言語 (基本)	上位優先		整数　0.33 ［μs］ 実数　0.40 ［μs］
機能	2 つの入力数値を比較し大きい数値を出力します。 型が違っても演算できます。ただし、整数は実数に変換後、実数演算します。		

D1 → D3
D2

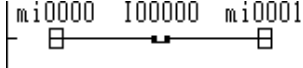
D1 ＞ D2　　なら　　D3 = D1

D1 <= D2　　なら　　D3 = D2

使用例

レジスタ mi0000 のデータとレジスタ kr0000 のデータ 100.0 が比較され、大きい方のデータがレジスタ mr0001 にストアされます。

レジスタ mi0000 のデータは整数ですが、レジスタ kr0000 のデータが実数ですので整数/実数の型変換を行ってから比較されます。

レジスタ kr0000 のデータ(100.0)を下限値とするリミッタになります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	下位優先		整数　0.40 ［μs］ 実数　0.28 ［μs］
機能	2つの入力数値を比較し小さい数値を出力します。 型が違っても演算できます。ただし、整数は実数に変換後、実数演算します。		

D1　D3

D2

D1 ＞ D2　なら　D3 = D2

D1 <= D2　なら　D3 = D1

使用例

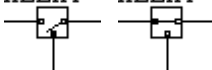
mi0000　mr0001

kr0000

100.0

レジスタmi0000のデータとレジスタkr0000のデータ100.0が比較され、小さい方のデータがレジスタmr0001にストアされます。

レジスタmi0000のデータは整数ですが、レジスタkr0000のデータが実数ですので整数/実数の型変換を行ってから比較されます。

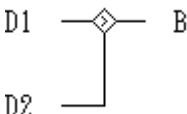
レジスタkr0000のデータ(100.0)を上限値とするリミッタになります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	数値積		0.33 ［μs]
機能	2 つの入力数値の論理積演算を行い、結果を出力します。		

注） 整数の演算のみ有効です。

使用例

レジスタ mi0000 のデータとレジスタ ki0001 のデータ(3)の論理積演算が行われ、結果がレジスタ mi0001 にストアされます。

レジスタ mi0000 のデータが(10)ならばレジスタ mi0001 には(2)がストアされます。

mi0000	0000	0000	0000	1010	(10)
ki0000	0000	0000	0000	0011	(3)
mi0001	0000	0000	0000	0010	(2)

種類	名称	シンボル	実行時間
データフロー言語（基本）	数値和		0.32 ［μs］
機能	2 つの入力数値の論理和演算を行い、結果を出力します。		

注） 整数の演算のみ有効です。

使用例

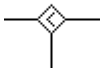
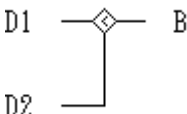
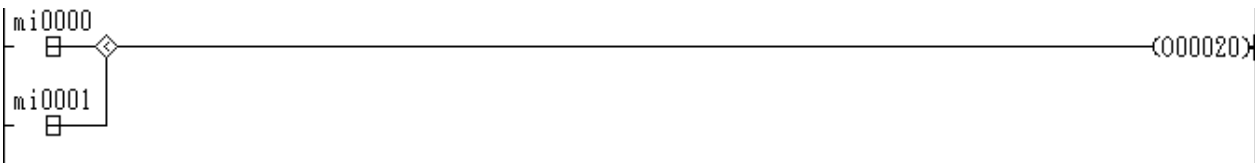
mi0000 mi0001

ki0001

3

レジスタ mi0000 のデータとレジスタ ki0001 のデータ(3)の論理和演算が行われ、結果がレジスタ mi0001 にストアされます。

レジスタ mi0000 のデータが(10)ならばレジスタ mi0001 には(11)がストアされます。

mi0000	0000	0000	0000	1010	(10)
ki0000	0000	0000	0000	0011	(3)
mi0001	0000	0000	0000	1011	(11)

種類	名称	シンボル	実行時間
データフロー言語 （基本）	数値排他和		0.31 ［μs］
機能	2 つの入力数値の排他的論理和演算を行い、結果を出力します。		

D3 = D1 ^ D2

注） 整数の演算のみ有効です。

使用例

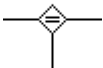
mi0000 mi0001

ki0001

3

レジスタ mi0000 のデータとレジスタ ki0001 のデータ(3)の排他的論理積演算が行われ、結果がレジスタ mi0001 にストアされます。

レジスタ mi0000 のデータが(10)ならばレジスタ mi0001 には(9)がストアされます。

mi0000	0000	0000	0000	1010	(10)
ki0000	0000	0000	0000	0011	(3)
mi0001	0000	0000	0000	1001	(9)

種類	名称	シンボル	実行時間
データフロー言語 （基本）	a 接点	RELAY ─┤├─	整数　0.34 ［μs］ 実数　0.36 ［μs］
機能	RELAY が ON ならば入力の数値を出力します。 OFF ならば出力の数値を 0 にします。		

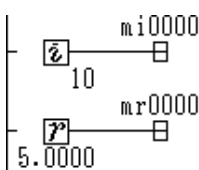
D1 ─(RELAY)─ D2

RELAY=ON　なら　D2 = D1

RELAY=OFF　なら　D2 = 0

使用例

mi0000　I00000　mi0001

リレーI00000 が ON の時は、レジスタ mi0000 のデータがレジスタ mi0001 にストアされます。
リレーI00000 が OFF の時は、レジスタ mi0001 に(0)がストアされます。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	b 接点	RELAY	整数　0.50 ［μs］ 実数　0.38 ［μs］
機能	RELAY が OFF ならば入力の数値を出力します。 ON ならば出力の数値を 0 にします。		

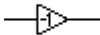
D1 —RELAY— D2

RELAY=ON　なら　D2 = 0

RELAY=OFF　なら　D2 = D1

使用例

mi0000　I00000　mi0001

リレーI00000 が OFF の時は、レジスタ mi0000 のデータがレジスタ mi0001 にストアされます。
リレーI00000 が ON の時は、レジスタ mi0001 に(0)がストアされます。

種類	名称	シンボル	実行時間
データフロー言語 (基本)	c 接点	RELAY RELAY	整数 0.39 ［μs］ 実数 0.33 ［μs］
機能	RELAY の論理値により 2 つの入力数値の一方を選択し出力します。		

RELAY=ON　なら　D3 = D1

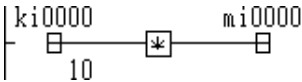
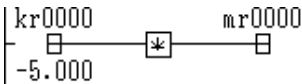
RELAY=OFF　なら　D3 = D2

RELAY=ON　なら　D3 = D2

RELAY=OFF　なら　D3 = D1

使用例

リレーI00000 が OFF の時はレジスタ mi0001 のデータが選択されてレジスタ mi0002 にストアされます。
リレーI00000 が ON の時はレジスタ mi0000 のデータが選択されてレジスタ mi0002 にストアされます。

リレーI00000 が OFF の時はレジスタ ki0000 のデータ(3)が選択されてレジスタ mi0003 にストアされます。
リレーI00000 が ON の時はレジスタ ki0001 のデータ(6)が選択されてレジスタ mi0003 にストアされます。

種類	名称	シンボル	実行時間
データフロー言語 (基本)	コンペア・ハイ		整数 0.17 [μs] 実数 0.13 [μs]
機能	2 つの入力数値の比較を行い、判定結果を論理値で出力します。		

D1 ＞ 2 なら B = ON

D1 <= D2 なら B = OFF

使用例

レジスタ mi0000 のデータがレジスタ mi0001 のデータよりも大きければリレーO00020 が ON になります。
これ以外ではリレーO00020 が OFF になります。

論理反転と組み合わせて、論理を変更することができます。
レジスタ mi0002 のデータがレジスタ mi0003 のデータと同じかレジスタ mi0003 のデータよりも小さければリレーO00021 が ON になります。
これ以外ではリレーO00021 が OFF になります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	コンペア・ロウ		整数　0.17 ［μs］ 実数　0.13 ［μs］

機能	2 つの入力数値の比較を行い、判定結果を論理値で出力します。

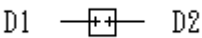
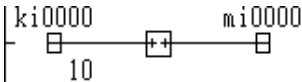
D1 ＜ D2　　なら　B = ON

D1 ＞= D2　　なら　B = OFF

使用例

mi0000
mi0001
(O00020)

レジスタ mi0000 のデータがレジスタ mi0001 のデータよりも小さければリレーO00020 が ON になります。
これ以外ではリレーO00020 が OFF になります。

論理反転と組み合わせて、論理を変更することができます。
レジスタ mi0002 のデータがレジスタ mi0003 のデータと同じかレジスタ mi0003 のデータよりも大きければリレーO00021 が ON になります。
これ以外ではリレーO00021 が OFF になります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	コンペア・イコール		整数　0.18 ［μs］ 実数　0.11 ［μs］
機能	2 つの入力数値を比較し、判定結果を論理値で出力します。		

D1 = D2　なら　B = ON

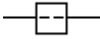
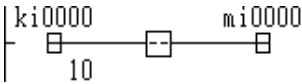
D1 ≠ D2　なら　B = OF

注） 使用するレジスタが実数の場合、表れない細かな数値のため ON しない場合があります。

使用例

mi0000
(O00020)
mi0001

レジスタ mi0000 のデータがレジスタ mi0001 のデータと同じならばリレーO00020 が ON になります。
これ以外ではリレーO00020 が OFF になります。

論理反転と組み合わせて、論理を変更することができます。
レジスタ mi0002 のデータがレジスタ mi0003 のデータと等しくないならばリレーO00021 が ON になります。
これ以外ではリレーO00021 が OFF になります。

種類	名称	シンボル	実行時間
データフロー言語 （基本）	ロード 局所定数 （整数、実数）	i— r—	整数 0.24 ［μs］ 実数 0.21 ［μs］
機能	局所的な定数（整数、実数）をロードします。		

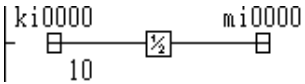
定数はプログラム内（パラメータでなく）に確保されます。

ロード局所定数（整数）はi形式のみの演算ブロック中でのみ使用できます。

1 つの演算ブロック内で（整数）と（実数）の混在はできません。

使用例

```
     mi0000
 i ─────⊟
   10
     mr0000
 r ─────⊟
5.0000
```

レジスタ mi0000 には整数値(10)がロードされます。
レジスタ mr0000 には実数値(5.0000)がロードされます。

種類	名称	シンボル	実行時間
データフロー言語 (関数 1)	符号変換		整数 0.18 [μs] 実数 0.12 [μs]
機能	入力数値の正負符号の反転を行い出力します。		

D1　　D2　　　　D2 = - (D1)

使用例

ki0000　　mi0000
-10

レジスタ ki0000 のデータ(−10)が正数に符号変換されてレジスタ mi0000 に(10)がストアされます。

kr0000　　mr0000
5.0000

レジスタ kr0000 のデータ(5.0000)が負数に符号変換されてレジスタ mr0000 に(−5.0000)がストアされます。

種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	絶対値変換	─[*]─	整数　0.30 ［μs］ 実数　0.20 ［μs］
機能	入力数値の絶対値をとり出力します。		

D1 ─[*]─ D2

D1 ＜ 0　　なら　D2 = −(D1)

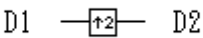
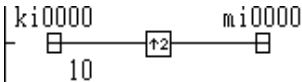
D1 ＞= 0　　なら　D2 =　D1

使用例

```
|ki0000          mi0000
├ ⊟────[*]────⊟
|  10
```

レジスタ ki0000 のデータ(10)が絶対値変換されてレジスタ mi0000 に(10)がストアされます。

```
|kr0000          mr0000
├ ⊟────[*]────⊟
|-5.000
```

レジスタ kr0000 のデータ(−5.0000)が絶対値変換されてレジスタ mr0000 に(5.0000)がストアされます。

種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	1‘補数	─[∿∿]─	0.19 ［μs］

機能	入力数値の補数演算を行い、結果を出力します。

D1 ─[∿∿]─ D2

D2 ＝ NOT （D1）

注） 整数演算のみ有効です。

使用例

mi0000 ─[∿∿]─ mi0001

レジスタ mi0000 のデータに対して 1 の補数演算を行い、結果をレジスタ mi0001 にストアします。
レジスタ mi0000 のデータが(10)ならば、レジスタ mi0002 に(−11)をストアします。

mi0000	0000	0000	0000	1010	(10)
mi0001	1111	1111	1111	0101	(-11)

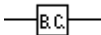
種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	インクリメント	─[++]─	整数　0.19 ［μs］ 実数　0.14 ［μs］
機能	入力数値に 1 を加算して結果を出力します。		

D1 ─[++]─ D2

D2　=　D1　+　1
(　D2　=　D1 + +　)

使用例

```
|ki0000              mi0000
├ ⊟────────[++]────────⊟
|   10
```

レジスタ ki0000 のデータ(10)に(1)を加算し、演算結果(11)をレジスタ mi0000 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 (関数 1)	デクリメント	─[--]─	整数　0.23 ［μs］ 実数　0.16 ［μs］
機能	入力数値から 1 を減算して結果を出力します。		

D1 ─[--]─ D2

D2 = D1 - 1
(D2 = D1 - -)

使用例

```
|ki0000            mi0000
├ ⊟──────[--]──────⊟
|   10
```

レジスタ ki0000 のデータ(10)から(1)を減算し、演算結果(9)をレジスタ mi0000 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 (関数 1)	2 分の 1	─[½]─	0.20 [μs]

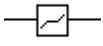
機能　入力数値の 2 分の 1 倍した結果を出力します。

D1 ─[½]─ D2　　　　D2 = D1 / 2

注) 整数演算のみ有効です。

使用例

```
|ki0000        mi0000
├ ⊟───[½]───⊟
|  10
```

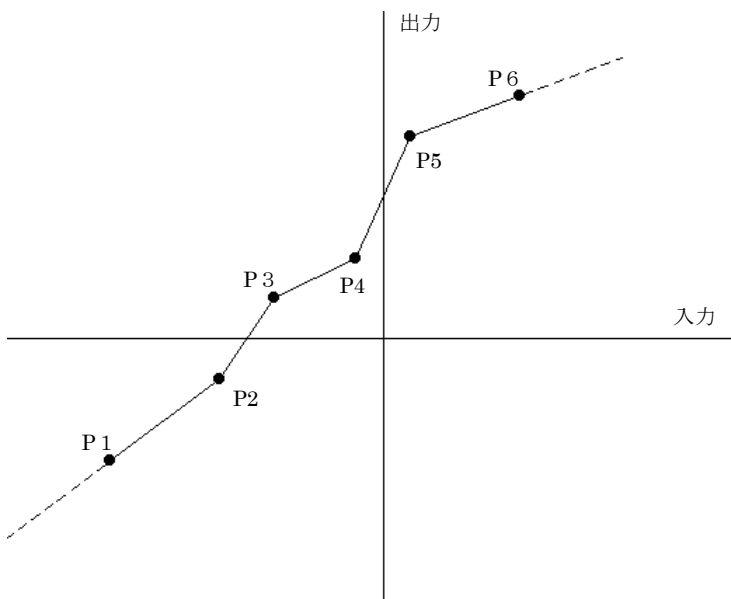
レジスタ ki0000 のデータ(10)を 2 分の 1 にし、演算結果(5)をレジスタ mi0000 にストアします。
この命令は整数レジスタのデータを符号付で 1/2 倍する場合に使用します。

種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	2 倍	─[×2]─	0.17 ［μs］
機能	入力数値の 2 倍した結果を出力します。		

D1 ─[×2]─ D2　　　D2 = D1 * 2

注） 整数演算のみ有効です。

使用例

```
|ki0000          mi0000
├ ⊟──────[×2]──────⊟
|  10
```

レジスタ ki0000 のデータ(10)を 2 倍にし、演算結果(20)をレジスタ mi0000 にストアします。
この命令は整数レジスタのデータを符号付で 2 倍する場合に使用します。

種類	名称	シンボル	実行時間
データフロー言語 (関数 1)	二乗	─[↑2]─	整数　0.25 ［μs］ 実数　0.14 ［μs］
機能	入力数値の二乗した結果を出力します。		

D1 ─[↑2]─ D2

D2 = D1 ** 2

(D2 = $D1^2$)

使用例

```
|ki0000              mi0000
├ ⊟────────[↑2]────────⊟
|   10
```

レジスタ ki0000 のデータ(10)を 2 乗し、演算結果(100)をレジスタ mi0000 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	平方根	─[√]─	整数　0.36 ［μs］ 実数　0.33 ［μs］
機能	入力数値の平方根を出力します。		

D1 ─[√]─ D2　　　　D2 ＝ SQRT （D1）

注） 入力値が負の値の場合、出力も負の値となります。

使用例

```
|ki0000          mi0000
├ ⊟─────[√]─────⊟
|   9
```

レジスタ ki0000 のデータ(9)の平方根を演算し、演算結果(3)をレジスタ mi0000 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	指数関数	─[↑N]─	3.60 ［μs］
機能	入力数値を指数演算し結果を出力します。		

D1 ─[↑N]─ D2 （上部：D3）

D2 = D3 ** D1

($D2 = D3^{D1}$)

注） 実数演算のみ有効です。

使用例

```
|kr0000   kr0001   mr0000
├─ ⊟────────[↑N]────────⊟
|4.0000   3.0000
```

レジスタ kr0000 のデータ(4.0000)に対して、レジスタ kr0001(3.0000)のデータで指数演算を行い、演算結果(64)をレジスタ mr0000 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 （関数 1）	ビットカウント	─B.C─	0.73 ［μs］
機能	入力数値を 16 ビット 2 進数として読み込み、ON しているビットの数を出力します。		

D1 ─B.C─ D2

注） 整数演算のみ有効です。

使用例

```
|ki0000              mi0000
├ ⊟───────B.C───────⊟
| 1234
```

レジスタ ki0000 のデータ(1234)を 16 ビット 2 進数として読み込み、ON しているビット(1 になっている) の数を計算し、演算結果(5)をレジスタ mi0000 にストアします。

ki0000	0000	0001	1010	1010	(1234)
mi0001	0	+ 1	+ 2	+ 2 =	

種類	名称	シンボル	実行時間
データフロー言語 (関数 1)	グレイコード バイナリー	─[G.B]─	16.1 [μs]

機能	入力数値(グレイコード)を変換し、2 進数で結果を出力します。

グレイコードは、数値が 1 つ変化するのに対して、1 ビットしか変化しないため、位置決め制御などで使います。

D1 ─[G.B]─ D2

0～15 までのビットパターンは下記のようになります。

D2	D1	D2	D1	D2	D1	D2	D1
整数	グレイ	整数	グレイ	整数	グレイ	整数	グレイ
0000	0000	0100	0110	1000	1100	1100	1010
0001	0001	0101	0111	1001	1101	1101	1011
0010	0011	0110	0101	1010	1111	1110	1001
0011	0010	0111	0100	1011	1110	1111	1000

注） 整数演算のみ有効です。

使用例

mi0000 ─[G.B]─ mi0001

レジスタ mi0000 のデータをグレイコード変換して、演算結果を mi0001 にストアします。
レジスタ mi0000 のデータが(10)ならば、レジスタ mi0001 に(12)をストアします。

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	不感帯	─[⟋]─	整数　0.27 ［μs］ 実数　0.21 ［μs］

機能	入力数値が不感帯範囲内なら 0 を出力します。 入力数値が範囲外なら不感帯値(絶対値)を減算し、結果を出力します。

D3
D1 ─[⟋]─ D2

−D3 < D1 < +D3	なら	D2 = 0
+D3 <= D1	なら	D2 = D1 − D3
−D3 >= D1	なら	D2 = D1 + D3

使用例

```
mi0000  ki0000  mi0001
 ─□─────[⟋]─────□
          10
```

レジスタ mi0000 のデータがレジスタ ki0000 の符号変換したデータ(−10)より大きく、正数データ(10)より小さい場合には、(0)をレジスタ mi0001 にストアします。

レジスタ mi0000 のデータがレジスタ ki0000 のデータ(10)と等しいか大きい場合には、レジスタ mi0000 のデータからレジスタ ki0000 のデータ(10)を減算した結果をレジスタ mi0001 にストアします。

レジスタ mi0000 のデータがレジスタ ki0000 の符号変換したデータ(−10)と等しいか小さい場合には、レジスタ mi0000 のデータとレジスタ ki0000 のデータ(−10)を加算した結果をレジスタ mi0001 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	パターン		整数　1.70 ［μs］ 実数　1.50 ［μs］
機能	入力数値をパターンメモリにより折れ線近似変換し、結果を出力します。		

パターンデータはあらかじめツールのパターンデータで設定しておきます。

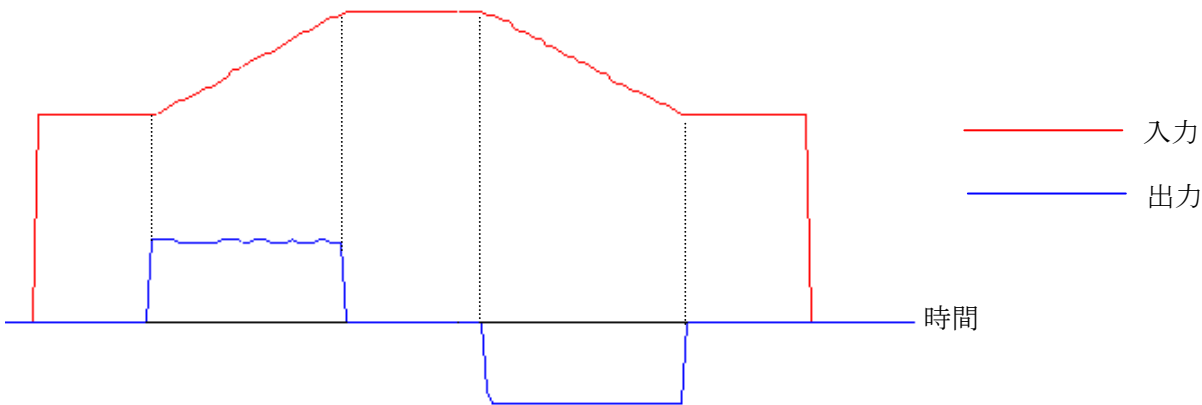
横軸のデータは必ず小さいほうから大きいほうに順に並べてください。

横軸は関数の入力値に相当し、パターンデータから離脱したデータが入力された場合でもパターンデータの傾斜で延長され、変換し出力します。

グラフ

入力が P1 より小さい場合は、直線 P1･P2 で延伸された近似直線に変換し出力します。
大きい場合も同様に直線 P5･P6 で延伸された近似直線に変換し出力します。

	入力	出力
P1/Q	−10	−3
P2/Q2	−6	−1
P3/Q3	−4	1
P4/Q4	−1	2
P5/Q5	1	5
P6/Q6	5	6

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	微分補償		1.40 [μs]
機能	入力数値の時間微分値の 3 回平均をとり、結果を出力します。		

関数引数設定内容

①微分ゲイン：秒単位系での微分係数（入力変化が毎秒 1.0 の時、1.0 を出力）

急激な変化量に対しては安全のため平均化しています。

演算パラメータは krxxxx 以外に mrxxxx も使用可能ですが、この場合各パラメータをユーザプログラムでセットしてください。

注） 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

微分補償

微分ゲイン	kr0000	10.000

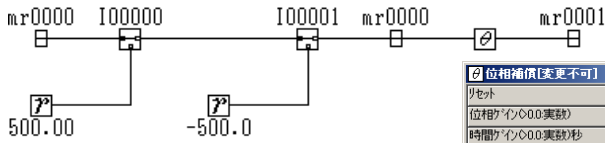
入力値が一定（傾き 0）のところでは、微分値も 0 なので出力が 0 となります。
入力値が常に変化している部分しか出力値も変化しません。

注） 下のトレンドグラフには急激な変化分についてはグラフ上で現れていません。

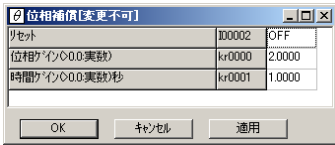
種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	位相補償	─[θ]─	1.47 [μs]

機能 入力数値に対して位相補償を行い、結果を出力します。

関数引数設定内容
　①リセット:入出力短絡リセット動作を指令します。
　②位相ゲイン(A):1.0 との大小関係で進み位相または遅れ位相となります。
　③時間ゲイン(T):秒単位での時間係数

　演算パラメータは krxxxx 以外に mrxxxx も使用可能ですが、この場合各パラメータをユーザプログラムでセットしてください。

　リセットを ON すると入出力間を短絡しますので、任意の値にプリセットすることができます。

　注) 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。
時間ゲインにより出力値が入力値に近づいていくカーブの大きさが変化します。
小さければ小さな弧を描き、大きければ大きな弧を描きます。

スキャンタイム:10ms
トレンドサンプリングタイム:100ms
での例

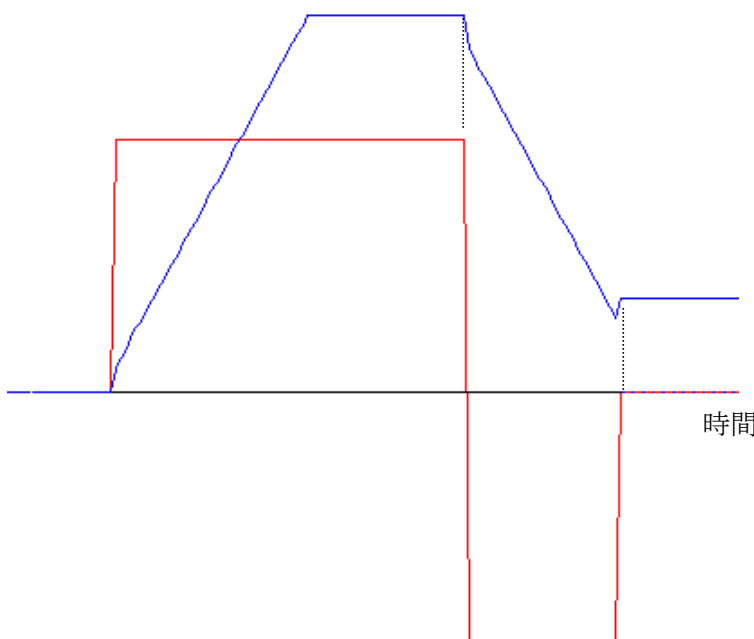
位相ゲイン(A) < 時間ゲイン(T)

リセット	G00000	
位相ゲイン(A_1)	kr0000	0.5000
時間ゲイン	kr0001	1.0000

位相ゲイン(A) > 時間ゲイン(T)

リセット	G00000	
位相ゲイン(A_1)	kr0000	2.0000
時間ゲイン	kr0001	0.5000

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	PI 補償		2.53 [μs]
機能	入力数値に対してPI補償(比例・積分)を行い、結果を出力します。		

関数引数設定内容
①リセット:入出力短絡リセット動作を指令します。
②ホールド:積分ホールドSW(積分停止)
③比例ゲイン:
④積分ゲイン:秒単位系での積分係数
⑤上限値:出力する上限値を指定します。
⑥下限値:出力する下限値を指定します。

演算パラメータはkrxxxx以外にmrxxxxも使用可能ですが、この場合各パラメータをユーザプログラムでセットしてください。

リセットをONすると入出力間を短絡しますので、任意の値にプリセットすることができます。
注) 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。
比例ゲインによって始めの出力値の高さが変化し、積分ゲインによって出力値の傾きが変化します。

PI 補償

リセット	G00000	
ホールド	G00001	
比例ゲイン	kr0000	0.1000
積分ゲイン	kr0001	3.0000
上限値	kr0002	30.000
下限値	kr0003	-30.000

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	直線形変化率制限		1.08 [μs]
機能	入力数値の時間的変化率制限を行い、結果を出力します。		

関数引数設定内容
①リセット：入出力短絡リセット動作を指令します。
②最大上昇率(>0.0：正の値)：毎秒当たりの出力上昇率制限値
(例：10.0＝毎秒 10 以下の上昇を許可)
③最大下降率(<0.0：負の値)：毎秒当たりの出力下降率制限値
(例：-10.0＝毎秒 10 以下の下降を許可)

演算パラメータは krxxxx 以外に mrxxxx も使用可能ですが、この場合各パラメータをユーザプログラムでセットしてください。

リセットを ON すると入出力間を短絡しますので、任意の値にプリセットすることができます。
注) 実数演算のみ有効です。

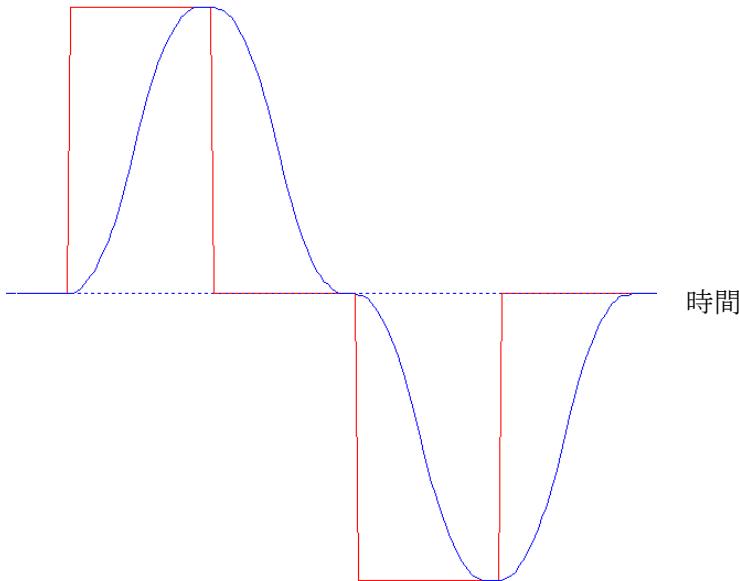
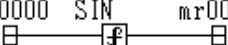
グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。
上昇・下降率によって出力値の傾きを設定できます。(ステップ入力を加えた場合)

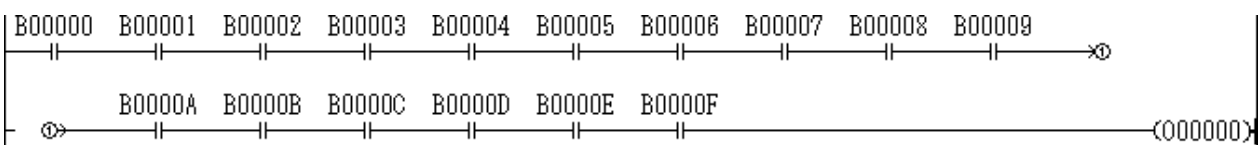
直線変化率制限

リセット	G00000	
最大上昇率	kr0000	0.1000
最大下降率	kr0001	-0.1000

種類	名称	シンボル	実行時間
データフロー言語 （関数 2）	S 字形変化率制限 （S-ARC）		4.01 ［μs］
機能	入力数値に対しＳ字時間的変化率制限を行い、結果を出力します。		

関数引数設定内容
①リセット：入出力短絡リセット動作を指令します。
②最大上昇率（>0.0）：毎秒当たりの出力上昇率制限値
③最大下降率（<0.0）：毎秒当たりの出力下降率制限値
④加・上昇率（>0.0）：加速開始時の毎秒当たりの加加速度値
⑤減・上昇率（<0.0）：加速終了時の毎秒当たりの減加速度値
⑥減・減少率（>0.0）：減速終了時の毎秒当たりの減減速度値
⑦加・減少率（<0.0）：減速開始時の毎秒当たりの加減速度値
⑧S 字速終了係数(>0.0)：加減速終了時の変化率制限値
通常は④～⑦の絶対値の一番大きい値の 2 倍位で設定する。

リセットを ON すると入出力間を短絡しますので、任意の値にプリセットすることができます。
注） 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。
ARC と同様ですが、直線になる前のカーブ（B1～4）も設定しますので、S 字形のような波形を出力します。
（注意） 加減速中に入力値を変更するとオーバシュートする場合があります。

S 字型変化率制限

リセット	G00000	
最大上昇率	kr0000	10.000
最大下降率	kr0001	−10.000
加・上昇率	kr0002	0.020
減・上昇率	kr0003	−0.020
減・減少率	kr0004	0.0020
加・減少率	kr0005	−0.0020
S 字速終了係数	kr0006	0.0040

時間

入力
出力

種類	名称	シンボル	実行時間
データフロー言語 （関数 3）	三角関数 逆三角関数		SIN 7.4［μs］ COS 7.1［μs］ TAN 7.3［μs］ ASIN 7.2［μs］ ACOS 7.2［μs］ ATAN 9.3［μs］
機能	入力数値に対して三角（逆三角）関数演算を行い、結果を出力します。		

sin 関数	D1 —SIN[f]— D2	D2 ＝ sin（D1）
cos 関数	D1 —COS[f]— D2	D2 ＝ cos（D1）
tan 関数	D1 —TAN[f]— D2	D2 ＝ tan（D1）
asin 関数	D1 —ASIN[f]— D2	D2 ＝ $\sin^{-1}$（D1）
acos 関数	D1 —ACOS[f]— D2	D2 ＝ $\cos^{-1}$（D1）
atan 関数	D1 —ATAN[f]— D2	D2 ＝ $\tan^{-1}$（D1）

注） 実数演算のみ有効です。

使用例

mr0000 —SIN[f]— mr0001

mr0001 ＝ SIN(mr0000)

レジスタ mr0000 のデータの正弦を演算し、結果をレジスタ mr0001 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 （関数 2）	無条件サブルーチン	XXXXXX ─Sb─	6.64 ［μs］
機能	サブルーチンを無条件で実行します。		

シンボルをダブルクリックすると引数設定画面が出てユーザがサブルーチンに対して引数を設定することができます。

サブルーチン内ではスタックレジスタ（sr0000、si0000、SI0000）を使用してデータの受け渡しを行います。スタックレジスタには引数設定画面から設定します。

実際には以下の様なデータの流れになります。

入力データ		スタックレジスタ		出力データ
整数データ	→	si0000	→	整数データ
実数データ	→	sr0000	→	実数データ
リレー/コイル	→	SI0000	→	リレー／コイル
呼び出し元		サブルーチン		呼び出し元

使用例

無条件でサブルーチン AAAA を実行します。レジスタ mi0000、mi0001 は慣例的に使用しますが、このデータも使用したい時には、レジスタ mi0000 のデータはサブルーチン AAAA のスタックレジスタ si0000 に渡されます。そしてサブルーチン AAAA で計算されたデータがスタックレジスタ si0000 にストアされたら、レジスタ mi0001 にデータがストアされます。

ただし、サブルーチン AAAA で使用しない場合には、レジスタ mi0000 のデータがレジスタ mi0001 にストアされます。

種類	名称	シンボル	実行時間
LD 言語	ジャンプ命令	─(JPXXXX)┤	――――――――
	ラベル命令	XXXXXX [L]─	
機能	ジャンプ：指定回路、指定ラベルへジャンプします。 ラベル：ジャンプ先ラベルに使用します。		

論理回路の 1 つとみなされます。

XXXX は回路番号またはラベル名（4 桁）です。

注 1） サブプログラム・サブルーチン間でのジャンプはできません。
注 2） また 1 ヵ所でループするようプログラムも作成可能ですが、永久ループにならないようにしてください。
注 3） ラベルの右側にはレジスタのストアにしてください。

使用例

```
B00000
─┤├──────────────────────────────(JPABCD)┤

kr0000  mr0000
─ ⊟────⊟
10.000
 ABCD   mi0000
─ [L]────⊟
```

リレーB00000 が ON の時は、ラベル ABCD ラインにジャンプし、ラベル ABCD との間にあるプログラムを実行しません。

リレーB00000 が OFF の時は、レジスタ kr0000 のデータ(10.000)がレジスタ mr0000 にストアされ、レジスタ mi0000 には 1 がストアされます。
リレーB00000 が ON の時は、レジスタ kr0000 のデータ(10.000)がレジスタ mr0000 にストアされず、レジスタ mi0000 には 0 がストアされます。

種類	名称	シンボル	実行時間
LD 言語	結合子 (ストア)	──→①	0.16 ［μs］
	結合子 (ロード)	①>──	0.16 ［μs］

機能	論理演算、数値演算結果の中間メモリのストアおよびロードを行います。

直列に 12 個以上論理記号、数値記号がある時に使用します。

必ずネットワークとネットワークの間に入れてください。

1 回路に 10 組までシンボルが入れられますが、必ずストアの後にロードしてください。

使用例

種類	名称	シンボル	実行時間
LD 言語	サブルーチン プログラム処理終了	─(RETURN)┤	2.00 ［μs］
機能	サブルーチンプログラムを終了します。		

サブルーチンプログラム内で、ある条件で終了させたいときに使用します。

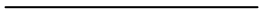
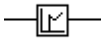
使用例

```
g00000  AAAAAA  g00001
─□──────[Sb]──────□        呼び出し元プログラム

si0002  si0008
─□──────□                  サブルーチンプログラム
SI0040
─┤├──────────────────────────────────(RETURN)
si0006  si000A
─□──────□
```

リレーI00000 が OFF の時は、スタックレジスタ si0002 のデータ=ki0000(5)のデータがスタックレジスタ si0008 にストアされ、レジスタ mi0000 にデータ(5)がロードされます。スタックレジスタ si0006 のデータ=z00009 のデータはスタックレジスタ si000A にストアされ、レジスタ mi0001 にロードされます。

しかし、リレーI00000 が ON の時は、スタックレジスタ si0002 のデータ(5)はそのままスタックレジスタ si0008 にストアされますが、スタックレジスタ si0006 のデータは I00000 が ON した時のデータが si000A にストアされたままになります。(z00009 は 1 ミリカウンタなので 100 の時 ON したなら si000A には 100 がストアされます。また I00000 を ON すれば、si0006 のデータがストアされます。)

引数名	ラベル名	値
si0002	ki0000	5
SI0040	I00000	
si0006	z00009	
si0008	mi0000	
si000A	mi0001	

種類	名称	シンボル	実行時間
データフロー言語 （関数 2）	（算術）平均	─[x̄]─	────────
機能	引数で設定した先頭アドレスから入力数値分のデータの算術平均値をとり、 結果を出力します。		

関数引数設定内容

① バッファアドレス先頭（mrXXXX）：入力が 1 より少ない場合は 1 とみなし 1 個目のデータの値を返します。

使用例

```
      mr0000
[r]─────□
10.000
      mr0001
[r]─────□
11.000
      mr0002
[r]─────□
12.000
      mr0003
[r]─────□
13.000
      mr0004
[r]─────□
14.000
kr0000          gr0000
 □──────[x̄]──────□
5.0000
```

算術平均の引数　バッファアドレス先頭：mr0000

以上のように設定すると、算術平均はレジスタ kr0000 のデータ(5.0000)と引数を読み込み、
(mr0000 + mr0001 + mr0002 + mr0003 + mr0004)/5
の演算結果(12.000)をレジスタ gr0000 にストアします。

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	フィルタ		2.42 [μs]
機能	入力数値に対し周波数制限を行い、結果を出力します。		

関数引数設定内容

①リセット:入出力短絡リセット動作を指令します。
②下限周波数(>0.0:正の値):3db 低下の下限周波数
③上限周波数(>0.0:正の値):3db 低下の上限周波数

注) 実数演算のみ有効です。

グラフ

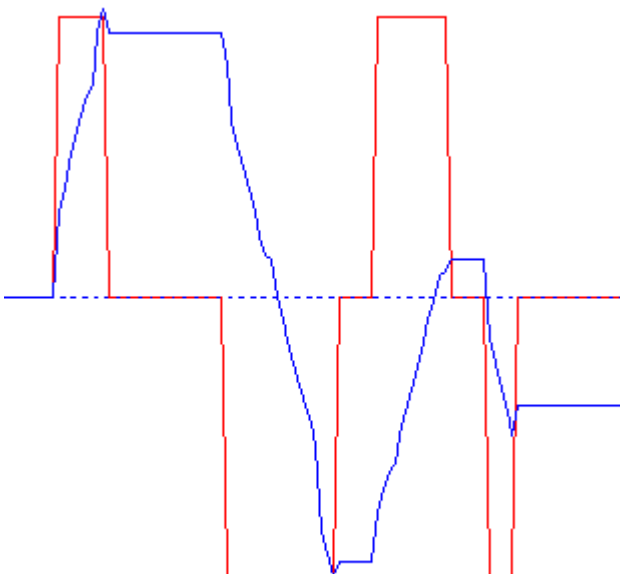
右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

フィルタ

リセット	G00000	
下限周波数	kr0000	0.0001
上限周波数	kr0001	0.0500

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	PID 補償		2.36 [μs]

機能	入力数値に対して PID 補償を行い、結果を出力します。

関数引数設定内容

①リセット:入出力短絡リセット動作を指令します。
②ホールド:積分停止 SW
③ゼロクリア:ゼロリセット指令するリレーを指定します。
④比例ゲイン:
⑤積分ゲイン:秒単位系での積分係数(出力値が入力値に到達するまでの時間:秒)
⑥微分ゲイン:秒単位系での微分係数(入力変化が毎秒 1.0 の時、1.0 を出力)
⑦MAX リミット:出力する上限値を指定します。
⑧MIN リミット:出力する下限値を指定します。

リセットを ON すると入出力間を短絡しますので、任意の値にプリセットすることができます。

注) 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

フィルタ

リセット	G00000	
ホールド	G00001	
ゼロクリア	G00002	
比例ゲイン	kr0000	0.1000
積分ゲイン	kr0001	3.0000
微分ゲイン	kr0002	0.0100
MAX リミット	kr0003	30.000
MIN リミット	kr0004	-30.000

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	一時遅れ		1.70 ［μs］
機能	入力数値に対し一時遅れ応答を出力します。		

関数引数設定内容
①リセット：入出力短絡リセット動作を指令します。
②時定数：T 秒

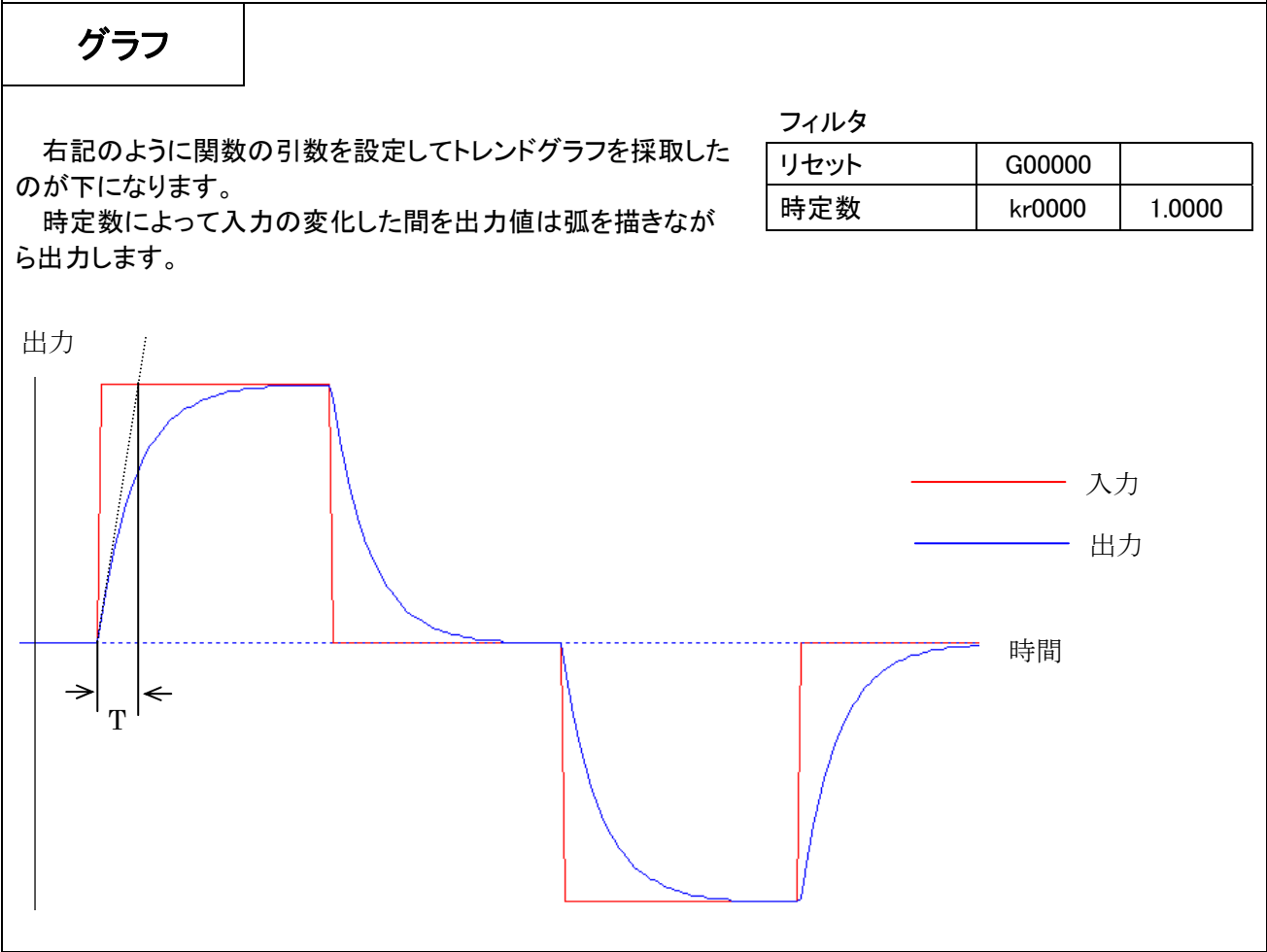
演算開始時に必ずリセット SW を ON してください。

注） 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。
時定数によって入力の変化した間を出力値は弧を描きながら出力します。

フィルタ

リセット	G00000	
時定数	kr0000	1.0000

種類	名称	シンボル	実行時間
データフロー言語 （関数 2）	ディレー （時間遅れ）	─⊡─	1.57 ［μs］
機能	入力数値に設定した遅れ時間を付加し出力します。		

関数引数設定内容

①リセット：入出力短絡リセット動作を指令します。

②遅れ時間：T（秒）

③サンプリング時間：⊿T（秒）　　サンプル数（T/⊿T）は 1000 以下で有効です。

リセット SW を ON するとディレーは無くなります。

注） 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

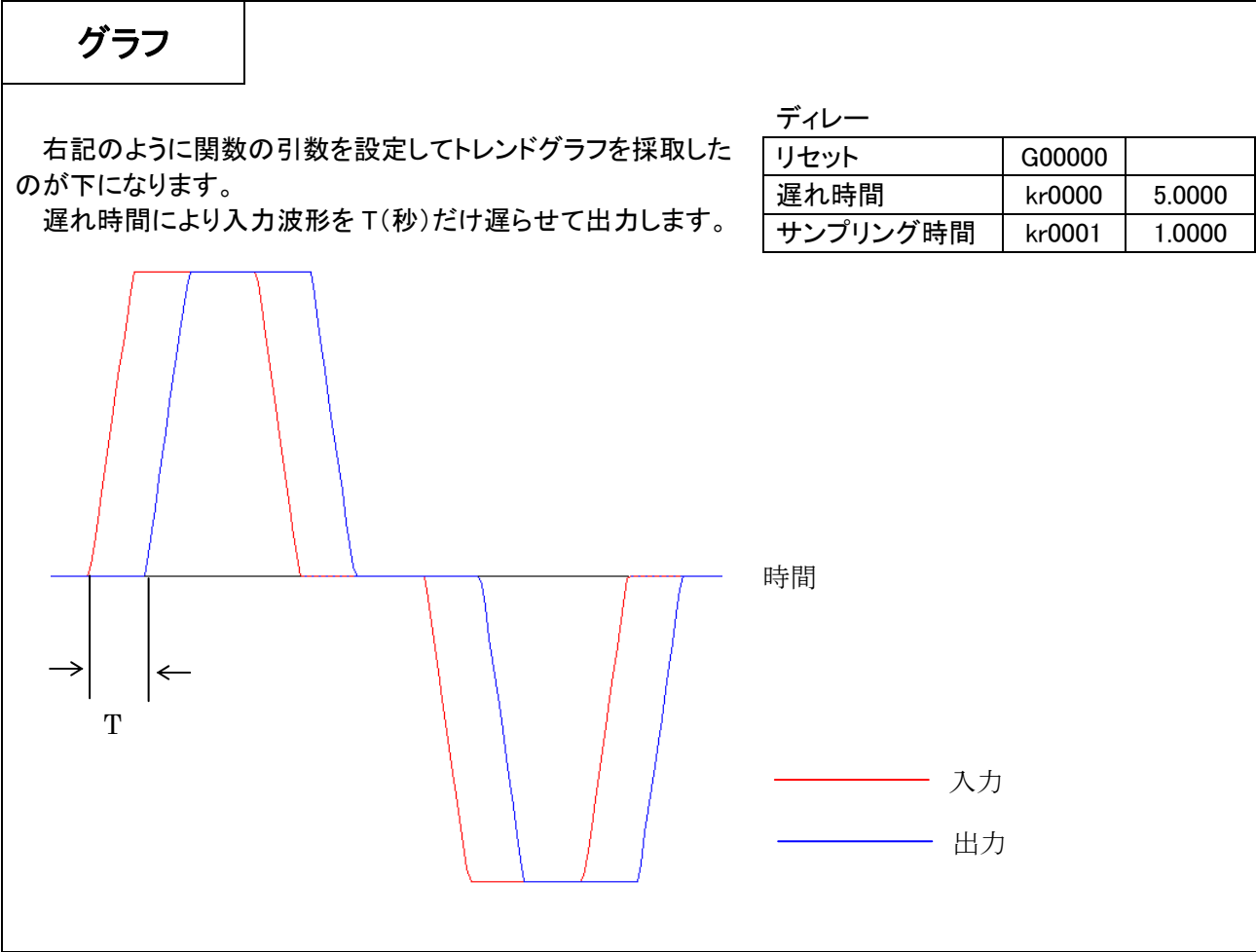
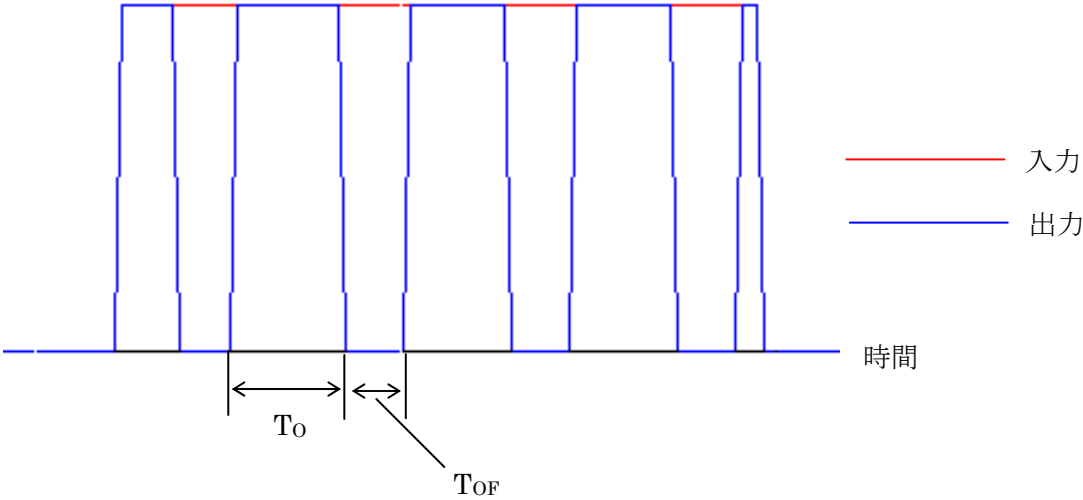
遅れ時間により入力波形を T（秒）だけ遅らせて出力します。

ディレー

リセット	G00000	
遅れ時間	kr0000	5.0000
サンプリング時間	kr0001	1.0000

時間

入力

出力

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	定周期パルス		1.39 [μs]
機能	入力数値を設定した時間でオン・オフさせ出力します。		

関数引数設定内容

①リセット：入出力短絡リセット動作を指令します。

②オン時間(秒)：出力を ON させる時間を指定します。

③オフ時間(秒)：出力を OFF させる時間を指定します。

注）実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

入力された波形をオン・オフ時間によって出力します。

定周期パルス

リセット	G00000	
オン時間	kr0000	5.0000
オフ時間	kr0001	3.0000

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	変数設定パターン		2.00 ［μs］
機能	入力数値をパターンメモリにより折れ線近似変換し出力します。		

関数引数設定内容

①ポイント数(≧2:整数):入力パターン数

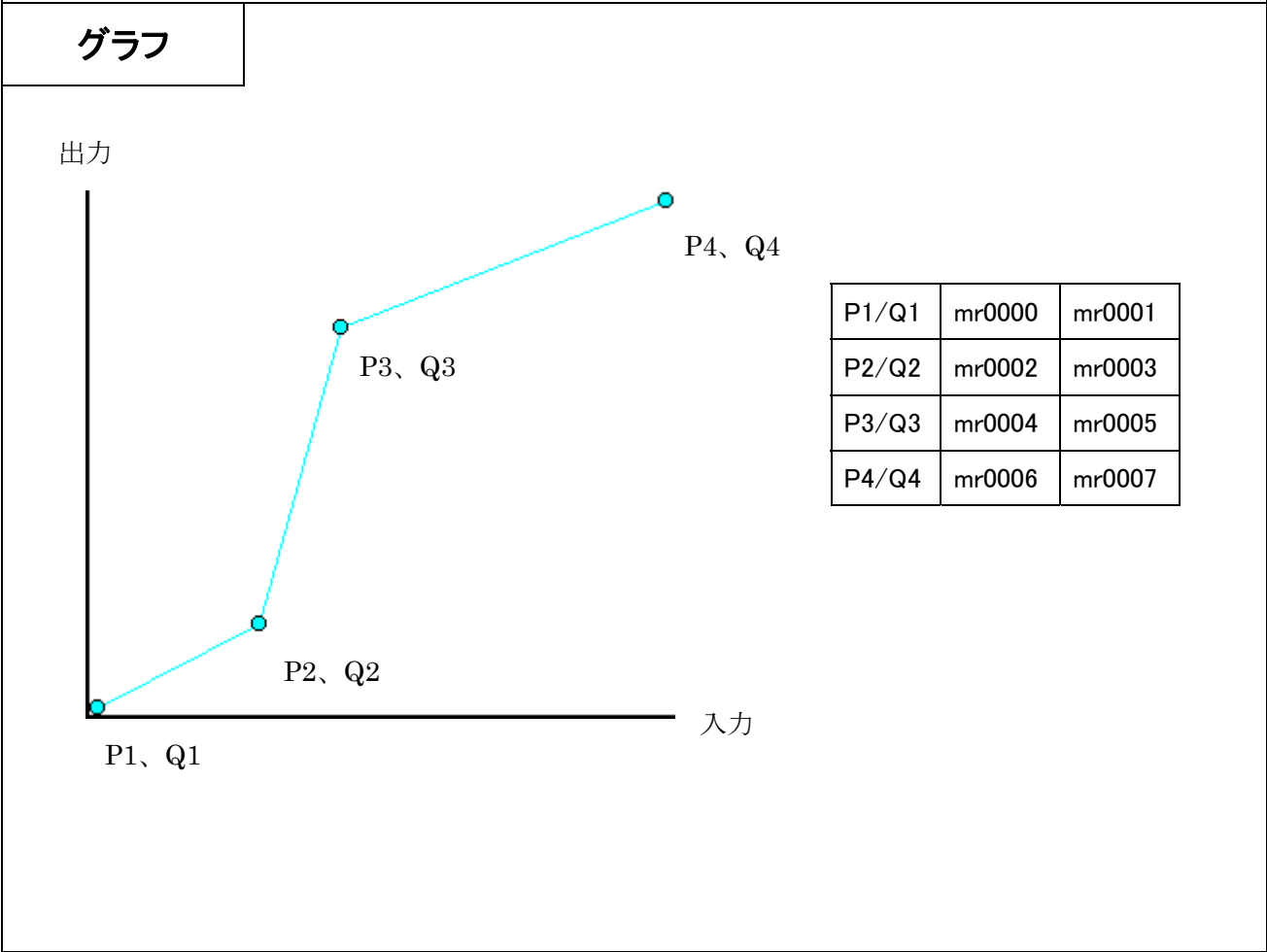
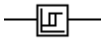
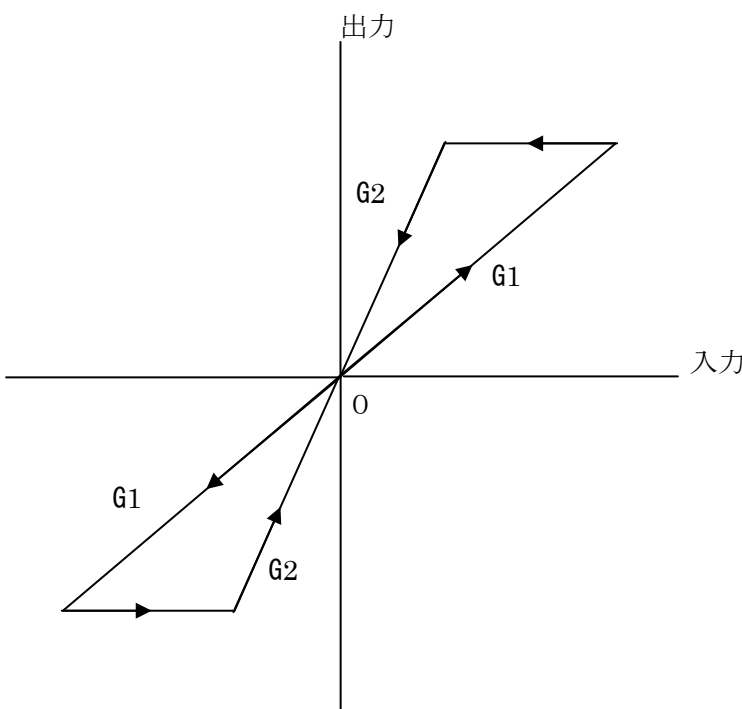
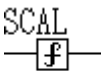
②パターンバッファ先頭(mrXXXX):入力バッファ先頭アドレス

パターンではあらかじめパターンデータで初期値を設定しましたが、ここでは回路中の実数値を変更することができます。

プロセス制御で得られデータを蓄積することによって学習制御に応用できます。

注） 実数演算のみ有効です。

グラフ

P1/Q1	mr0000	mr0001
P2/Q2	mr0002	mr0003
P3/Q3	mr0004	mr0005
P4/Q4	mr0006	mr0007

種類	名称	シンボル	実行時間
データフロー言語 （関数 2）	上下限リミッタ		0.72 ［μs］
機能	入力数値に上下限リミッタを付加し出力します。		

関数引数設定内容

①上限値：出力の上限値を指定します。

②下限値：出力の下限値を指定します。

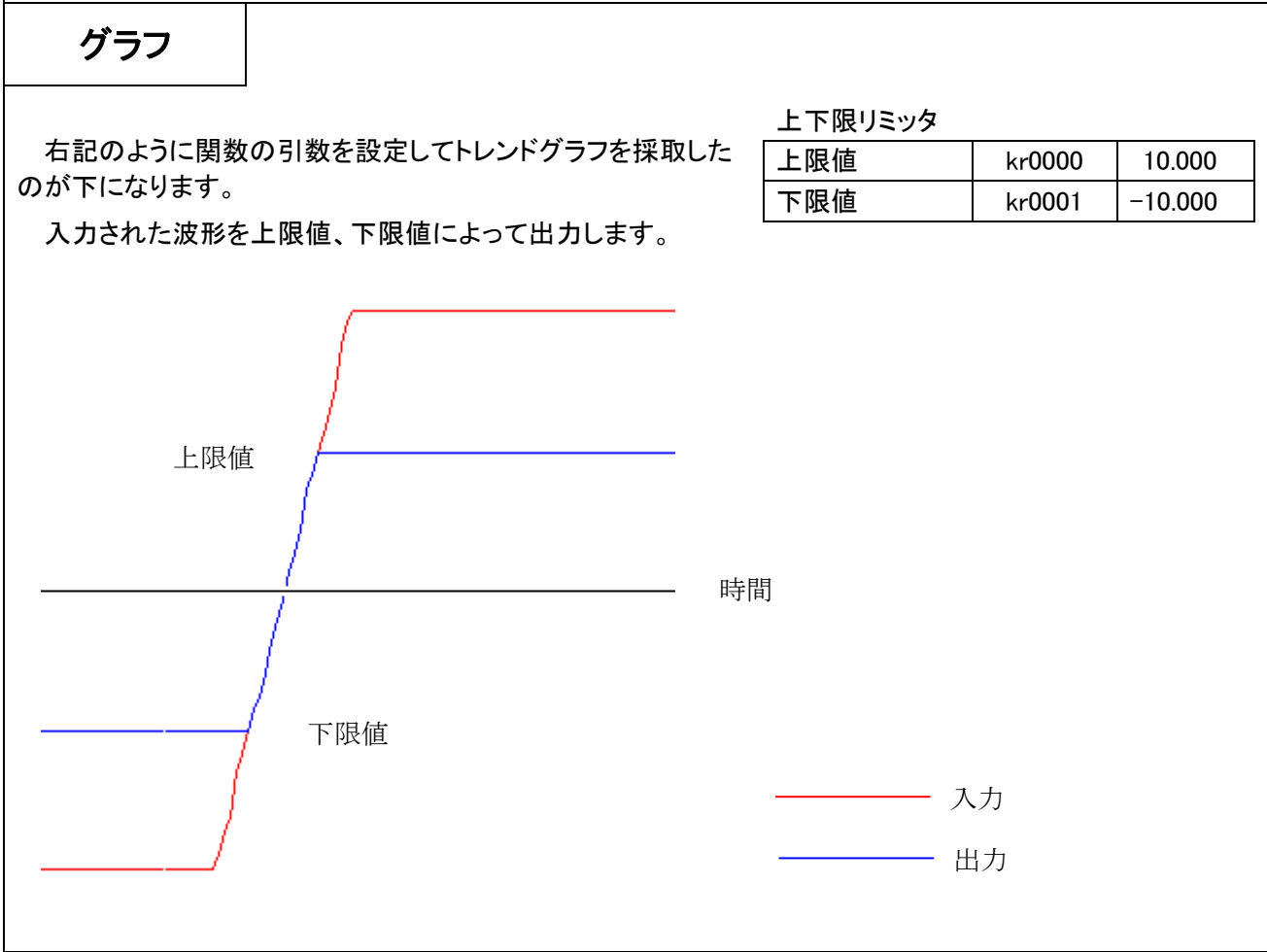
注） 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

入力された波形を上限値、下限値によって出力します。

上下限リミッタ

上限値	kr0000	10.000
下限値	kr0001	−10.000

種類	名称	シンボル	実行時間
データフロー言語 (関数 2)	ヒステリシス		1.27 ［μs］

機能	入力数値にヒステリシス(上昇下降時の 2 ゲイン増幅器)を付加し出力します。

関数引数設定内容:

①リセット:出力値 = 入力値 × G1 とします。

②ロウ側ゲイン:G1(0.0＜G1＜G2)

③ハイ側ゲイン:G2(0.0＜G1＜G2)

入力データが上昇時には G1 が有効となり下降時には G2 が有効となります。

上昇・下降または下降・上昇の切り替え時には一定の出力に留まります。

演算開始時に必ずリセット SW を ON してください。

注) 実数演算のみ有効です。

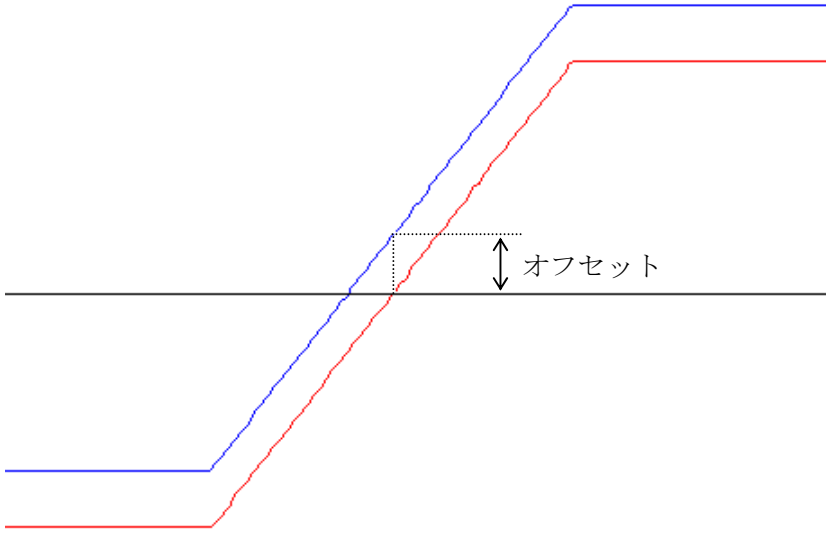
グラフ

入力データの変化の履歴により出力データが図に示すカーブとなります。

種類	名称	シンボル	実行時間
データフロー言語 （関数 3）	スケーリング	SCAL f	0.92 ［μs］
機能	入力数値をスケーリング（積和演算）を付加し出力します。		

関数引数設定内容：

①ゲイン：積和演算の乗算係数

②オフセット：積和演算の加算係数

出力 ＝ 入力 ＊ ゲイン ＋ オフセット

注） 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。
入力された波形をゲイン・オフセットによって出力します。

スケーリング

ゲイン	kr0000	1.0000
オフセット	kr0001	5.0000

種類	名称	シンボル	実行時間
データフロー言語 (関数 3)	バックラッシュ	BKLS ─[f]─	0.81 [μs]
機能	入力数値にバックラッシュ(一種の積分補償)を付加し出力します。		

関数引数設定内容

①リセット:入出力短絡リセット動作を指令します。

②バックラッシュの幅:W

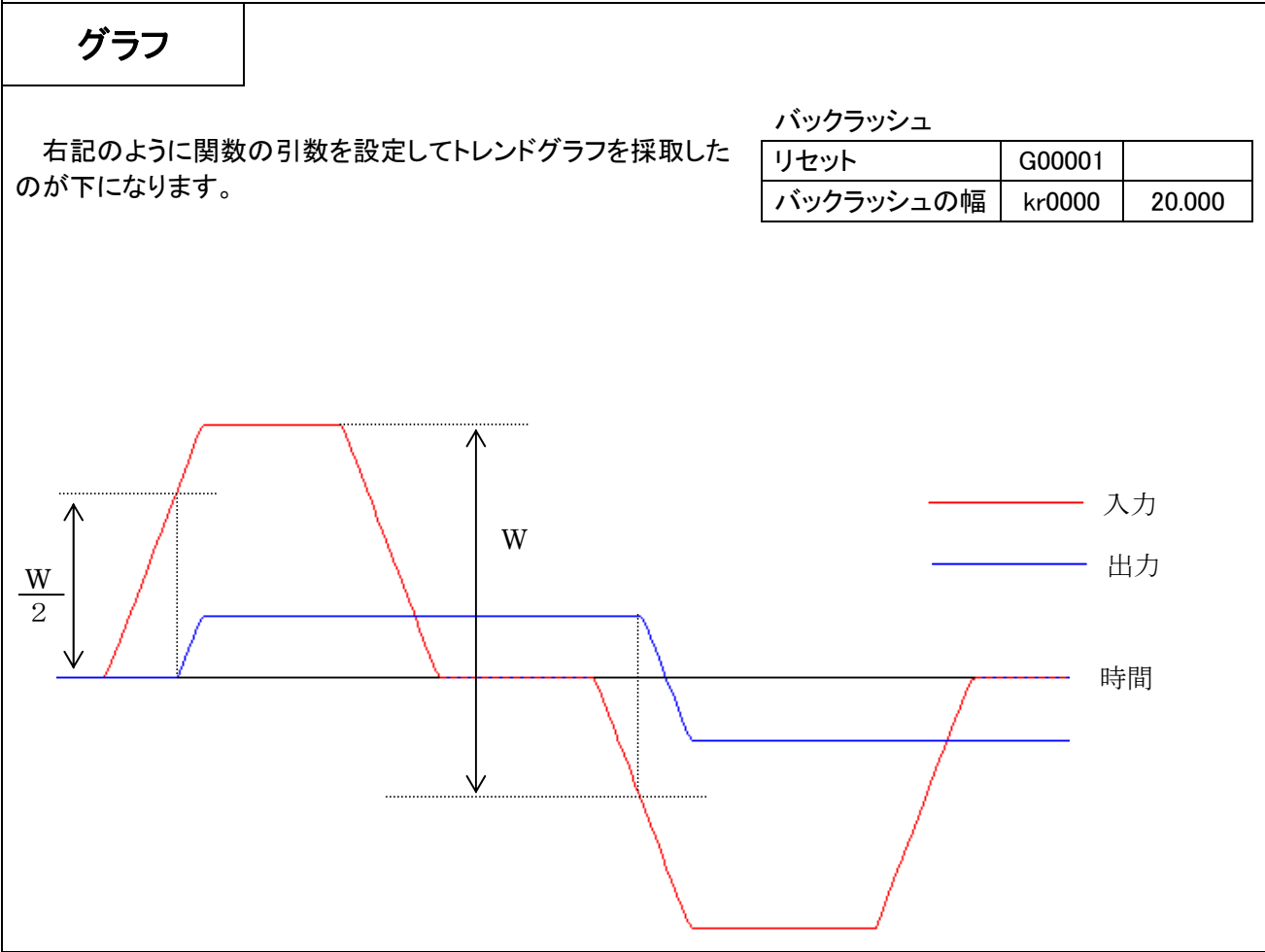
演算開始時に必ずリセット SW を ON してください。

注) 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

バックラッシュ

リセット	G00001	
バックラッシュの幅	kr0000	20.000

種類	名称	シンボル	実行時間
データフロー言語 (関数 3)	バックラッシュ補正	BKLC	0.95 [μs]
機能	入力数値をバックラッシュ補正(一種の微分補償)し出力します。		

関数引数設定内容

①リセット:入出力短絡リセット動作を指令します。

②バックラッシュの幅:W

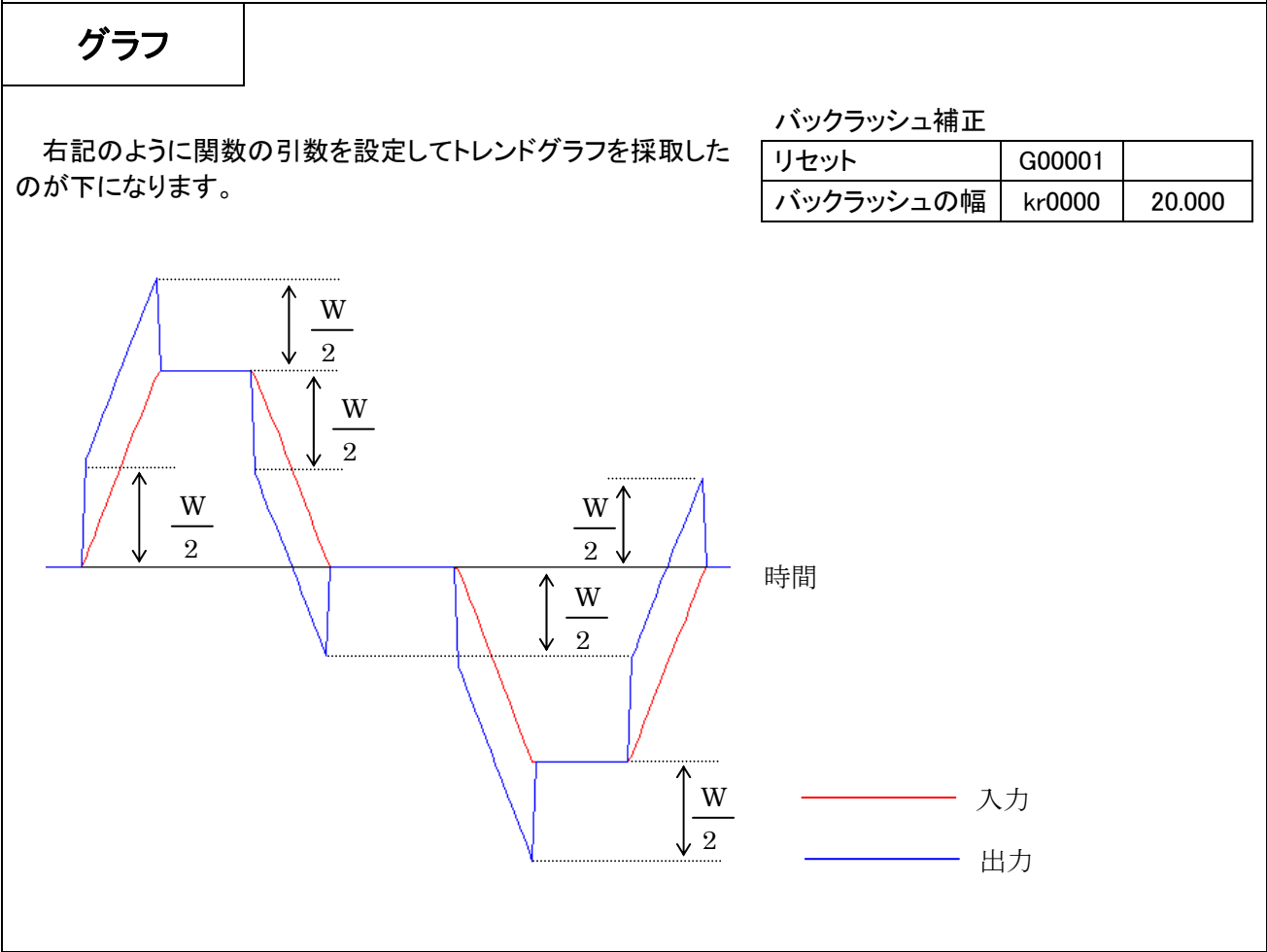
演算開始時に必ずリセット SW を ON してください。

注) 実数演算のみ有効です。

グラフ

右記のように関数の引数を設定してトレンドグラフを採取したのが下になります。

バックラッシュ補正

リセット	G00001	
バックラッシュの幅	kr0000	20.000

種類	名称	シンボル	実行時間
データフロー言語 （関数 2）	条件付サブルーチン	XXXXXX ─SB	―――――
機能	入力の論理条件によりサブルーチンを実行します。		

入力が ON の時サブルーチンを実行し、OFF の時は実行しません。
その他の内容は無条件サブルーチンと同様です。

使用例

```
B00000                                         AAAA
──┤├───────────────────────────────────────────SB
```

リレーB00000 が ON の時は、サブルーチン AAAA を実行します。
リレーB00000 が OFF の時は、サブルーチン AAAA を実行しません。

種類	名称	シンボル	実行時間
データフロー言語 （関数 3）	バイナリー グレイコード	BTOG	――――
機能	入力数値を整数データとして読み込み、グレイコード変換して出力します。		

D1 BTOG D2

注） グレイコード変換　　G.B :の操作を行います。間違えないでください。

使用例

mi0000 BTOG mi0001

レジスタ mi0000 のデータを 16 ビット整数として読み込み、グレイコードに変換し出力します。

レジスタ mi0000 のデータが(10)の場合、レジスタ mi0001 には(15)がストアされます

D1	D2	D1	D2	D1	D2	D1	D2
整数	グレイ	整数	グレイ	整数	グレイ	整数	グレイ
0000	0000	0100	0110	1000	1100	1100	1010
0001	0001	0101	0111	1001	1101	1101	1011
0010	0011	0110	0101	1010	1111	1110	1001
0011	0010	0111	0100	1011	1110	1111	1000

10 → 1010 → 1111 → 15

入力　　整数　　グレイコード　　出力

種類	名称	シンボル	実行時間
データフロー言語 (関数 3)	割り余り	DIVMOD	――――――
機能	入力値の除算値と余りを出力します。		

関数引数設定内容

① 除数（整数）: 入力値を割る数

② 余り（整数）: 余りをストアするレジスタ

使用例

mi0000 DIVMOD mi0001

DIVMOD

引数	ラベル	値
除数(整数)	ki0000	7
余り(整数)	mi0002	

右のように DIVMOD の引数を設定すると、レジスタ mi0000 のデータを割る数 ki0000(7)で割った余りがレジスタ mi0002 にストアされます。また、レジスタ mi0001 には商がストアされます。

レジスタ mi0000 のデータが(10)の場合、レジスタ mi0001 には商として(1)、レジスタ mi0002 には余りとして(3)がストアされます。

種類	名称	シンボル	実行時間
データフロー言語 （関数 3）	オンタイマ(TSTD)	TSTD ─[f]─	―――
	オフタイマ(TRTC)	TRTC ─[f]─	
機能	オンタイマリレー(TS、TD)とオフタイマリレー(TR、TC)を 1 行にまとめたもので 動作は同じです。		

TSTD：入力ビットが ON すると引数で設定した時間経過後にコイルが ON します。

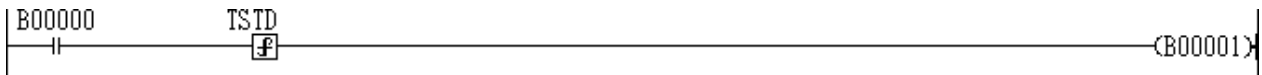
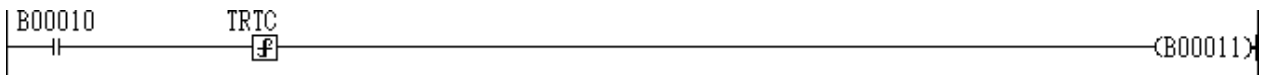
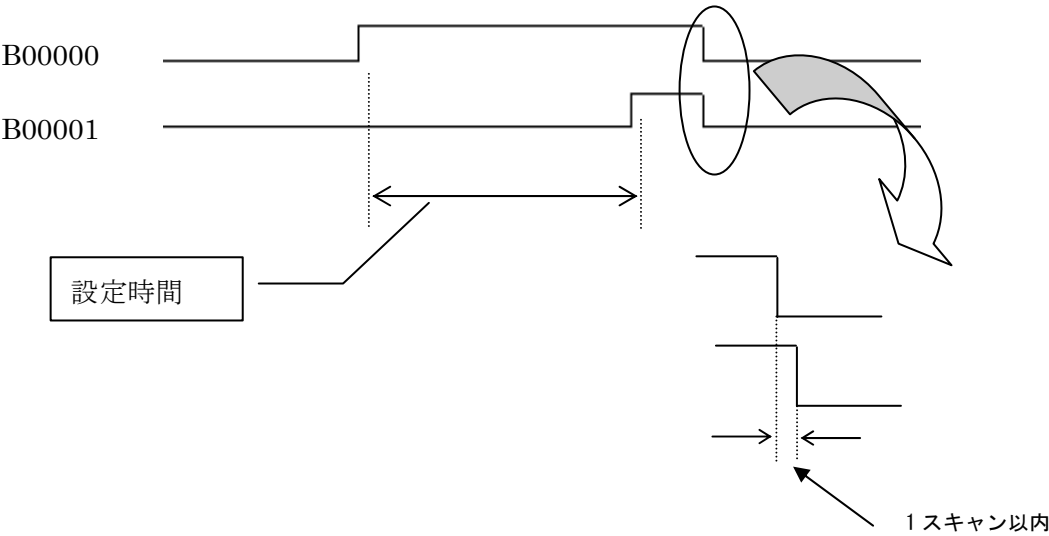
関数引数設定内容

① タイマ値（実数）：指定時間経過後に ON させる時間を設定します。

TRTC：入力ビットが OFF すると引数で設定した時間経過後にコイルが OFF します。

↓2 行書いていたところを 1 行で書けます。

関数引数設定内容

①タイマ値（実数）：指定時間経過後に OFF させる時間を設定します。

使用例
B00000
TSTD
B00001
リレーB00000 が ON してから TSTD で設定した時間経過後にリレーB00001 が ON します。
B00000
B00001
設定時間
1 スキャン以内
B00010
TRTC
B00011
リレーB00010 が OFF してから TRTC で設定した時間経過後にリレーB00011 が OFF します。
B00010
B00011
設定時間
1 スキャン以内

種類	名称	シンボル	実行時間
データフロー言語 （関数 3）	オン微分(USUC)	USUC —[f]—	――――
	オフ微分(DSDC)	DSDC —[f]—	
機能	オン微分リレー(US、UC)とオフ微分リレー(DS、DC)を 1 行にまとめたもので 動作は同じですが 1 スキャンは遅れません。		

USUC：入力ビットが ON すると 1 スキャン遅れずに 1 スキャン ON します。

```
B00000
─┤├──────────────────────(US0000)
UC0000
─┤├──────────────────────(B00001)
```

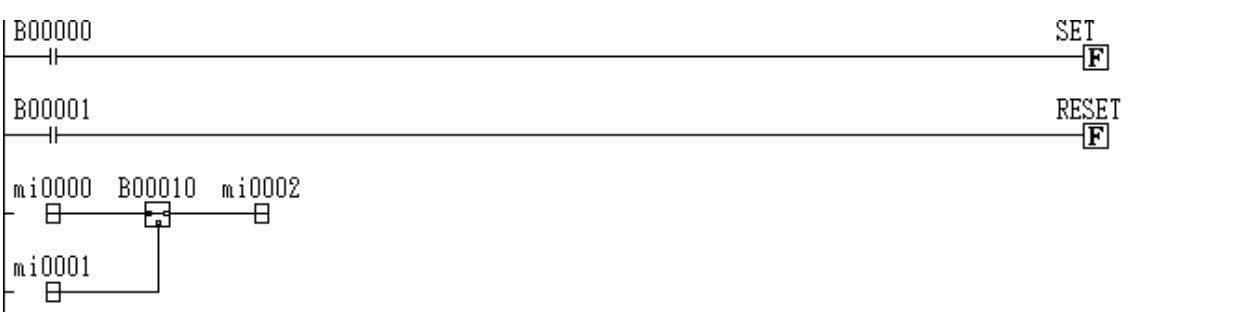
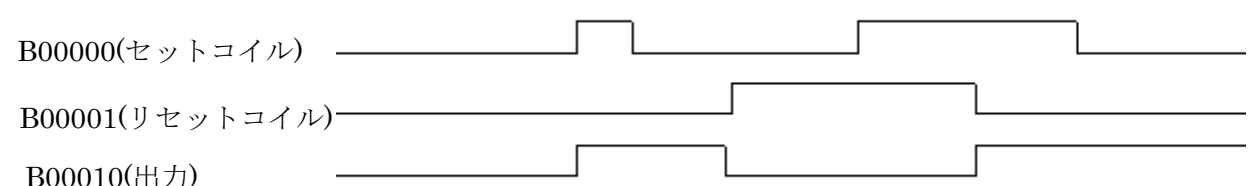
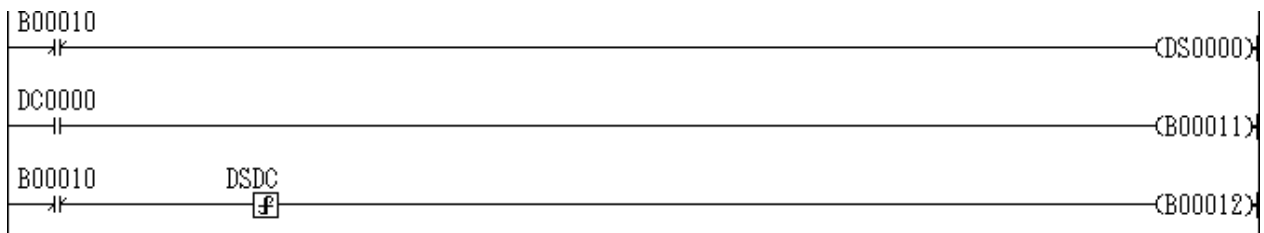
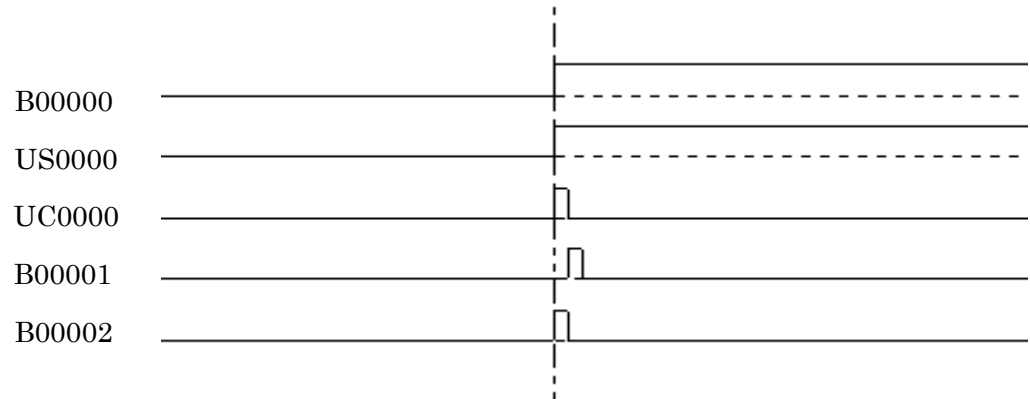
↓2 行で書いていたところを 1 行で書けます。

```
B00000      USUC
─┤├────────[f]──────────(B00001)
```

DSDC：入力ビットが OFF すると 1 スキャン遅れずに 1 スキャン ON します。

```
B00010
─┤├──────────────────────(DS0000)
DC0000
─┤├──────────────────────(B00011)
```

↓2 行で書いていたところを 1 行で書けます。

```
B00010      DSDC
─┤├────────[f]──────────(B00011)
```

B00000 が ON したら B00001 は 1 スキャン遅れて、1 スキャン分 ON しますが、
B00002 は 1 スキャン遅れず B00000 が ON したらすぐに 1 スキャン分 ON します。

B00010 が ON したら B00011 は 1 スキャン遅れて、1 スキャン分 ON しますが、
B00012 は 1 スキャン遅れず B00010 が ON したらすぐに 1 スキャン分 ON します。

種類	名称	シンボル	実行時間
データフロー言語 (関数 4)	セット(SET) リセット(RESET)	SET ─F　RESET ─F	――――
機能	セット：入力ビットが ON したら指定した出力ビットを ON し続けます。 リセット：入力ビットが OFF したら指定した出力ビットを OFF し続けます		

セット(SET)　：注)セットが ON している時は RESET が ON すると引数で設定した接点が OFF します。
関数引数設定内容
　①セットコイル：ON し続けるリレーを指定します。

リセット(RESET)：　注)リセットが ON している時は SET が ON しても引数で設定した接点は ON しません。
関数引数設定内容
　①リセットコイル：OFF し続けるリレーを指定します。

使用例

B00000　SET
B00001　RESET
mi0000　B00010　mi0002
mi0001

B00000=ON となると、B00010=ON となり、mi0002 には mi0001 の値がストアされます。
B00001=ON となると、B00010=OFF となり、mi0002 には mi0000 の値がストアされます。

B00000=ON となると、B00010=ON となります。(B00000=OFF となっても B00010=OFF とはなりません)
B00001=ON となると、B00010=OFF となります。(B00000=ON となっても B00010=ON とはなりません)
B00001=OFF となると、B00000=ON となっているので、B00010=ON となります。

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	カウンタ （UPDOWN）	UPDOWN —F	――――――
機能	カウンタ(NR、NP、NU、ND、NZ、n0)を 1 行にまとめたもので動作は同じです。		

関数引数設定内容

①リセットコイル：カウント現在値を 0 にするリレーを設定します。

②プロセットコイル：カウント現在値をカウントプリセット値で設定した値にするリレーを設定します。

③アップコイル：カウント現在値をインクリメントするにするリレーを設定します。

④ダウンコイル：カウント現在値をデクリメントするにするリレーを設定します。

⑤ゼロ検出接点：カウント現在値がゼロになったことを知らせるリレーを設定します。

⑥カウント現在値：現在値をストアするレジスタを設定します。

⑦カウントプリセット値：プリセットコイルを ON した時にカウント現在値にセットする値を設定します。

使用例

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	データ転送 （MOVW/MOVWD）	MOVW F　MOVWD F	――――

機能　指定したデータを指定したラベルへワード単位で転送します。

関数引数設定内容

①転送元ラベル：データを送信する先頭アドレスを指定します。

②転送先ラベル：データを受信する先頭アドレスを指定します。

③転送元オフセット：転送元ラベルの何番目からデータを送信するか指定します。（MOVW のみ）

④転送先オフセット：転送先ラベルの何番目からデータを受信するか指定します。（MOVW のみ）

⑤転送個数：送信するデータ個数を指定します。

使用例

B00000　MOVW F

右記のように設定すると mi000A から b00004 へデータを
5 ワード分転送します。

mi000A → b00004
mi000B → b00005
mi000C → b00006
mi000D → b00007
mi000E → b00008

MOVW

引数	ラベル	値
転送元ラベル	mi0000	
転送先ラベル	b00000	
転送元 オフセット	ki0000	10
転送先 オフセット	ki0001	4
転送個数	ki0002	5

種類	名称	シンボル	実行時間
データフロー言語 (関数 3)	整数変換	TODINT	――――
	実数変換	TOREAL	
機能	指定したデータを指定した型に変換し、結果を出力します。		

TODINT(実数入力を 32 ビット整数に変換)

関数引数設定内容

①転送先(2 点使用:偶数アドレス):入力実数データを 32 ビット整数に変換して出力するアドレスを指定します。

②転送先(2 点使用:偶数アドレス+1):入力実数データを 32 ビット整数に変換した符号を出力するアドレスを指定します。

TOREAL(32 ビット整数入力を実数に変換)

関数引数設定内容

①転送元(2 点使用:偶数アドレス):入力 32 ビット整数データを実数に変換して出力するアドレスを指定します。

②転送元(2 点使用:偶数アドレス+1):入力 32 ビット整数データを実数に変換した符号を出力するアドレスを指定します。

使用例

mr0000 TODINT mr0001

TODINT の場合、右記のように設定し、入力実数レジスタ mr0000 のデータが(-12.5600)の場合
mi0010 = -13 mi0011 = -1 となります。

TODINT

引数	ラベル	値
転送先 (偶数アドレス)	mi0010	
転送先 (偶数アドレス+1)	mi0011	

mr0010 TOREAL mr0011

TOREAL の場合、右記のように設定すると、

mr0011 = 131082 となります。

mr0011 = ki0000 + ki0001 ＊ 65536
= 10 + 2 ＊ 65536
= 10 + 131072
= 131082

TOREAL

引数	ラベル	値
転送元 (偶数アドレス)	ki0000	10
転送元 (偶数アドレス+1)	ki0001	2

種類	名称	シンボル	実行時間
データフロー言語 (関数 4)	チャンネルオープン	M_OPEN ─[f]─	─────────
機能	イーサネットにおいて、メッセージの通信相手の設定をするための関数です。この設定は次ページ以降で説明する M_SEND(メッセージ送信)、M_RECV(メッセージ受信)で使われます。		

関数引数設定内容

① 通信局番(スロット番号)

通信を行うイーサネットモジュール(CPUモジュール)のスロット番号を(0～9)を指定します。自CPUモジュールで通信を行う場合0となります。

② チャンネル番号:

通信モジュール内のチャンネル番号(コネクション番号:1～9)を指定します。

③ ステーション番号(L):通信相手のIPアドレス(下位 16 ビット)

④ ステーション番号(H):通信相手のIPアドレス(上位 16 ビット)

⑤ モジュール種別番号:0(未使用)

⑥ 通信モード:コネクションの通信条件を設定します。

⑦ 通信サブモード:(μGPCsH では未使用)

⑧ 通信相手ポート番号:通信相手のポート番号を設定します。

⑨ 自己ポート番号:自己のポート番号を設定します。

⑩ エラーフラグ:オープン処理が異常終了したとき 1 スキャンだけ ON します。

⑪ ステータス:エラー内容を表示します。

⑫ コネクション番号:オープン要求後、 H:スロット番号 L:チャンネル番号 が入ります。

<命令の動作>

① 入力リレー(B00000)の立ち上がり (OFF→ON) により通信局番(スロット番号)で指定されたモジュールのオープン処理が開始されます。(オープン処理は 1 スキャンでは終了しません)

② オープン処理が正常完了した場合正常フラグが ON となり、コネクション番号にコネクション番号が出力されます。この状態で M_SEND、M_RECV の使用ができるようになります。

③ オープン処理が正常に行われない場合は、エラーフラグが 1 スキャン ON となり、ステータスにエラーコードが出力されます。

④ 入力リレーを OFF にするとクローズ処理を行います(クローズ処理も 1 スキャンでは終了しません)。

⑤ クローズ処理が終了すると正常フラグが OFF になります(クローズ処理は異常終了することはありません)

＜命令の注意事項＞

① オープン方法は受信用の「Passive 方式」と送信用の「Active 方式」があります。通信を行うためには受信用のオープン処理、送信用のオープン処理があります。

② 送信するためには送信相手先が受信可能状態となっている必要があるため受信用の「Passive 方式」のオープン処理を先に完了しておく必要があります。

③ オープン中に入力リレーを ON → OFF にするとクローズ処理を行います。

④ クローズ処理の後、再オープンを行うときは、通信相手側を一旦クローズした後、再オープンの処理を行う必要があります。

<引数詳細>

1）ステーション番号(L)、(H)

通信相手先の IP アドレスを設定します。IP アドレスは、16 進数または 10 進数または 10 進数で設定します。下位 16 ビットをステーション番号(L)に、上位 16 ビットをステーション番号(H)に設定します。

例）IP アドレスが 172. 16. 0. 1 のときには、次のように設定します。

ステーション番号(L)＝0001(h)または1

ステーション番号(H)＝AC10(h)または-21488

2） 通信モード

チャンネルオープンにするコネクションの通信条件を、ビット情報として 1 ワードのデータにそれぞれ設定します。1 ワードの内容については次の通りです。

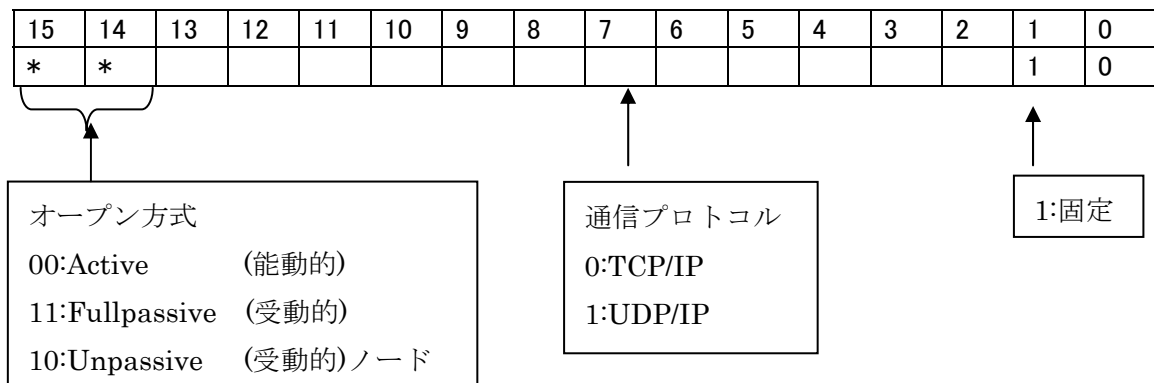
0082:UDP/IP

0002:TCP/IP 能動オープン

C002:TCP/IP 受動オープン

8002:TCP/IP 受動オープン

ビット詳細

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*													1	0

(a) 通信プロトコル
各コネクション別の通信プロトコルで、TCP/IP を使用するのか、UDP/IP を使用するのかを設定します。

(b) オープン方式
TCP/IP でオープンするときは、Fullpassive/Unpassive オープン(受動的オープン)するノードのオープン処理完了後に、Active オープン(能動的オープン)するノードのオープンを行います。
①Active オープン方式
TCP コネクションのオープン受動状態となっている他ノードに対して能動的なオープン処理を行います。
②Fullpassive オープン方式
　通信アドレス設定エリアに設定した特定ノードに対してのみ、TCP コネクションの受動的なオープン処理を行います。通信アドレス設定エリアに設定した他ノードからの Active なオープン要求待ち状態となります。
③Unpassive オープン方式
　ネットワークに接続されているすべての他ノードに対して、TCP コネクションの受動的なオープン処理を行います。ネットワーク内のすべての他ノードに対して、Active なオープン要求待ち状態となります。

3) エラーステータス

名称	コード	内容
パラメータ異常	177(B1h)	通信局番(スロット番号)で指定したスロットに、イーサネットモジュールが存在しない場合
チャンネルオープン異常	193(C1H)	通信モードの設定に異常な値を設定した場合
ポート指定異常	200(C8h)	ＩＰアドレスもしくは自己ポート番号、通信相手ポート番号に異常値が設定された場合

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	メッセージ送信	M_SEND	――――――
機能	M_OPEN で設定した通信相手にメッセージ送信を行います。		

関数引数設定内容

① コネクション番号：M_OPEN により解説したコネクション番号を設定します。

② 送信データ格納変数：送信データが格納されているデータサイズを設定します。

③ 送信データ格納変数サイズ：送信データが格納されているデータサイズを設定します。（ワード単位）

④ エラーフラグ：メッセージ送信が正常に行われなかったとき、1 スキャン ON します。

⑤ ステータス：メッセージ送信が正常に行われなかったとき、その内容を出力します。

＜命令の動作＞

①入力リレーの立ち上がり（OFF→ON）でコネクション番号に設定したコネクション番号のステーションへメッセージ送信を行います（送信処理は 1 スキャンでは終了しません）。

②メッセージ送信が正常に終了すると、正常フラグが 1 スキャンの間 ON します。

③メッセージ送信が正常に行われない場合は、エラーフラグ”が 1 スキャン ON となり、ステータスにエラーコードが出力されます。

＜命令の注意事項＞

①1 回のメッセージ送信で送信可能なデータ量は５１２ワードです。

②メッセージ送信中（リレー入力の立ち上がりから正常フラグまたはエラーフラグが立ち上がるまで）入力リレーは無効です。

③メッセージ送信中は送信データ格納変数を変更しないでください。変更した場合の送信データは保障されません。

④送信データ格納変数サイズで指定したデータ数が送信データ格納変数で指定した変数サイズを超過する場合、超過分のデータは不定となる場合があります。送信データ格納変数サイズには、必ず指定した変数のサイズを入力してください。

⑤入力リレーには M_OPEN の正常フラグが ON してから ON フラグが入力されるようにプログラムしてください。

<M_SEND 使用時の注意事項>

①UDP/IP の汎用通信モードでは送達確認およびフロー制御は行いません。受信側の受信処理が間に合わなくなった場合、受信バッファが一杯になり次に送られてきたデータは破棄されます。したがって、送信側の送信完了数と受信側の受信完了数は不一致になります。また受信バッファが一杯になった場合、バッファ解放に約 10 秒要するため、その間受信動作が停止することがあります。

②Full Passive オープンにて IP アドレス、ポート番号が一致しない相手からオープン要求を受信した場合、1度コネクション確立してから Full Passive 側が Active 側へクローズ要求を行います。そのため Active 側では、オープン正常完了してデータ送信を行ったときにエラーステータス C7h（強制クローズ）となります。

③送信側のポート番号が受信側と一致しない場合は、送信異常となり送信側から強制クローズが行われ、エラーステータス"C7h：（強制クローズ）"が発生します。

<エラーステータス>

名称	コード	内容
パラメータ異常	177(B1h)	通信局番（スロット番号）で指定したスロットに、イーサネットモジュールが存在しない場合
チャンネルオープン異常	193(C1H)	通信モードの設定に異常な値を設定した場合
ポート指定異常	200(C8h)	ＩＰアドレスもしくは自己ポート番号、通信相手ポート番号に異常値が設定された場合

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	メッセージ受信	M_RECV	————
機能	M_OPEN で設定した通信相手にメッセージ受信を行います。		

関数引数設定内容

①コネクション番号：M_OPEN により解説したコネクション番号を設定します。

②送信データ格納変数：送信データが格納されているデータサイズを設定します。

③送信データ格納変数サイズ：送信データが格納されているデータサイズを設定します。（ワード単位）

④エラーフラグ：メッセージ送信が正常に行われなかったとき、1 スキャン ON します。

⑥　ステータス：メッセージ送信が正常に行われなかったとき、その内容を出力します。

＜命令の動作＞

①入力リレーの立ち上がり（OFF→ON）でコネクション番号に設定したコネクション番号のステーションからメッセージ受信を行います（受信処理は 1 スキャンでは終了しません）。

②メッセージ受信が正常に終了すると、正常フラグが 1 スキャン ON します。

③メッセージ受信が正常に行われない場合は、“ERROR”が 1 スキャンの間“1”となり、“STATUS”にエラーコードが出力されます。

<命令の注意事項>

①1 回のメッセージ送信で送信可能なデータ量は５１２ワードです。

②メッセージ受信中（入力リレーの立ち上がりから正常フラグまたはエラーフラグが立ち上がるまで）入力リレーは ON に保持してください。入力リレーを OFF にすることは受信一時中断を意味します。

③受信一時中断後、入力リレーを立上げる（OFF→ON）と受信を再開します。このときにコネクション番号、受信データ格納変数、受信データ格納変数サイズを変更しても、中断前の入力値で再開します。変更はメッセージ受信処理には反映されません。

④メッセージ受信処理の終了後、次のスキャンでも入力リレーが ON に保持されていると、新たなメッセージ受信処理を開始します。

⑤受信処理中は受信データ格納変数を保持しておいてください。書き換えた場合の受信メッセージデータは保障されません。

⑥受信データ格納変数サイズで指定した数が受信データ格納変数で指定した変数のサイズを超過する場合、他の変数領域を書き換えてしまう場合があります。受信データ格納変数サイズには、必ず指定した変数サイズを入力してください。

⑦入力リレーには M_OPEN の正常フラグが ON になってから入力されるようにプログラムしてください。

＜M_RECV 使用時の注意事項＞

M_SEND と同様です。「M_SEND 使用時の注意事項」を参照してください。

＜エラーステータス＞

名称	コード	内容
パラメータ異常	177(B1h)	通信局番(スロット番号)で指定したスロットに、イーサネットモジュールが存在しない場合
チャンネルオープン異常	193(C1H)	通信モードの設定に異常な値を設定した場合
ポート指定異常	200(C8h)	ＩＰアドレスもしくは自己ポート番号、通信相手ポート番号に異常値が設定された場合
チャンネルクローズ	199(C7H)	通信相手局にクローズされた場合

種類	名称	シンボル	実行時間
データフロー言語 (関数 4)	マトリクス (MATRIX)	MATRIX —[f]—	————————
機能	マトリクス入力をする関数です。		

関数引数設定内容

①入力レジスタ:ストローブにより出力データが切り替わる外部機器を接続します。

②出力レジスタ:ストローブ出力(外部機器ストローブ入力に接続します。)

③マトリクス入力レジスタ先頭名:ストローブ出力により入力されたデータを順次格納していくレジスタ名の先頭を指定します。

使用例

入力レジスタ:i00000(1 ワード分のデータ入力用レジスタ名)

出力レジスタ:o00001(ストローブパルス発生用の出力レジスタ名)

マトリクス入力レジスタ先頭名:mi0010

o00001(O00010～O0001F)のストローブ出力により入力された i00000 データを mi0010 から mi001F へ順次格納します。

i00000=1　O00010=ON　mi0010=1
i00000=2　O00011=ON　mi0011=2
i00000=3　O00012=ON　mi0012=3
↓
i00000=16　O0001F=ON　mi001F=16
i00000=17　O00010=ON　mi0010=17
i00000=18　O00011=ON　mi0011=18

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	FREAD	FREAD	―――――――
機能	CFカードに保存されたファイルを読み出します。		

関数引数設定内容

① 属性(CSV 桁)

0：バイナリファイルとして読み出します。

0以外：CSVファイルとして読み出しそのCSVファイルの１行の桁数を指定します。

② ﾌｧｲﾙ名格納変数

ファイル名が格納された変数名を指定します。データはアスキーコードとします。

③ 読出ﾃﾞｰﾀ格納変数

読みだしたデータを格納する変数名を指定します。

④ 読出ﾃﾞｰﾀ格納変数ｻｲｽﾞ

読みだしたデータを格納する変数の使用可能領域を指定します。

⑤ ｴﾗｰﾌﾗｸﾞ

エラー発生時、ONします。

⑥ ｽﾃｰﾀｽ

エラー発生時、エラーコードが格納されます。

⑦ 読出ﾌｧｲﾙｻｲｽﾞ

読みだしたファイルのサイズ（ワード単位）が格納されます。

使用例

Z000E8がONすることにより、g00000～へ値を格納します。
実際の処理はバックグラウンドで行いますので読み出しが終了したら B00000 がONします。

ファイル名のアスキーコードの入力は定数データ入力のki変数域をダブルクリックすることにより、入力できます。

ステータス一覧
(1)ファイル名異常(コード:65)
ファイル名格納変数にファイル名として不正な文字が含まれています。
(2)ファイル処理中(コード:35)
他プログラム(他箇所)でファイル関数実行中です。

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	FWRITE	FWRITE	――――
機能	PLC内データをCFカードにファイルとして保存します。		

関数引数設定内容

① 属性(CSV 桁)

0:バイナリファイルとして読み出します。

0以外:CSVファイルとして読み出しそのCSVファイルの1行の桁数を指定します。

② ﾌｧｲﾙ名格納変数

ファイル名が格納された変数名を指定します。データはアスキーコードとします。

③書込ﾃﾞｰﾀ格納変数

書き込むデータが格納された変数名を指定します。

④書込ﾃﾞｰﾀ格納変数ｻｲｽﾞ

書き込むデータのサイズ（ワード単位）を格納します。

⑤ｴﾗｰﾌﾗｸﾞ

エラー発生時、ONします。

⑥ｽﾃｰﾀｽ

エラー発生時、エラーコードが格納されます。

使用例

Z000E8がONすることにより、g00000～のデータが格納されたファイルを生成します。
実際の処理はバックグラウンドで行いますので書き込みが終了したら B00000 がONします。

Z000E8 FWRITE (B00000) 書込完了

ファイル名のアスキーコードの入力は定数データ入力のki変数域をダブルクリックすることにより、入力できます。

ステータス一覧

（1）ファイル名異常（コード：65）

ファイル名格納変数にファイル名として不正な文字が含まれています。

（2）ファイル処理中（コード：35）

他プログラム（他箇所）でファイル関数実行中です。

（3）ファイルアクセス異常（コード：66）

ファイルアクセスにて異常が発生しました。

種類	名称	シンボル	実行時間
データフロー言語 （関数 4）	POKEKI	POKEKI	――――――――
機能	ki（定数データ）の値をアプリケーションプログラムとして保存します。 （リセット時は書き込んだデータとなります。）		

関数引数設定内容

① ki 先頭オフセット(整数)

書き込みたいki領域の先頭を指定します。

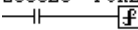
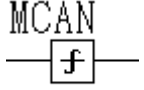
② 書込サイズ(整数)

書き込みたいkiのサイズ（ワード単位）を指定します。

③ 書込データ(整数)

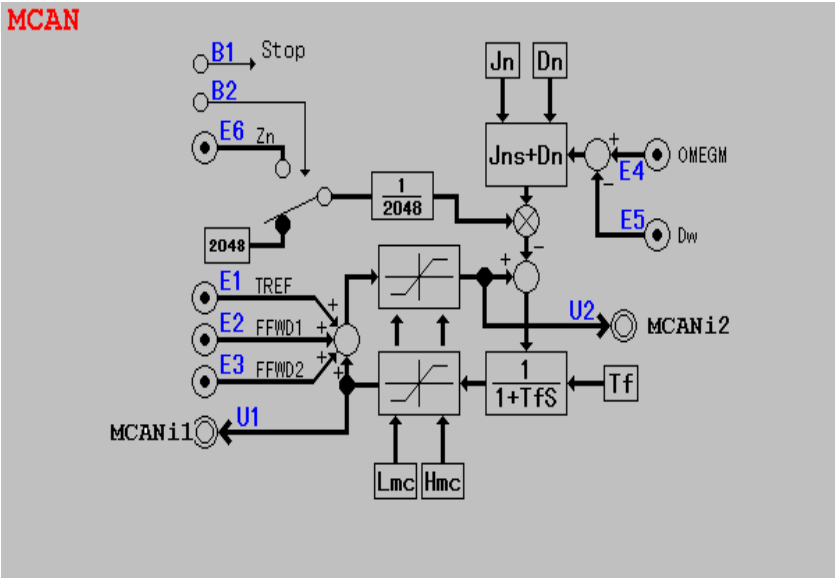
kiに書き込みたいデータが格納された変数を指定します。

使用例

Z000E8がONすることにより、ki0000～を1，2，3に変更します。
書き込み終了するとB00000がONします。

Z000E8　POKEKI　(B00000) 書込完了

種類	名称	シンボル	実行時間
データフロー言語 (関数 4)	汎用通信	C_FREE F	――――
機能	汎用通信用の関数です。		

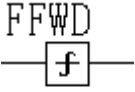
（1）関数引数設定内容	
0000	送信要求：データ送信を開始します。送信終了時にはアプリケーションで OFF する必要があります。
0001	送信データ長：送信するデータ長をバイト数で指定します。
0002	送信データアドレス：送信データの先頭アドレスを指定します。
0003	受信データアドレス：受信データの先頭アドレスを指定します。
0004	パラメータアドレス：ポート初期化用パラメータの先頭アドレスを指定します。
0005 (未使用) ～ 0006 (未使用)	
0007	送信完了：送信が完了すると ON します。(1 スキャン)
0008 (未使用) ～ 0009 (未使用)	
000A	受信完了：受信が完了すると ON します。(1 スキャン)
000B(未使用) ～ 000C(未使用)	
000D 000E	受信データ長：受信したデータ長が格納されます。 RS-485 局番：汎用通信モジュールの回線番号が格納されます。

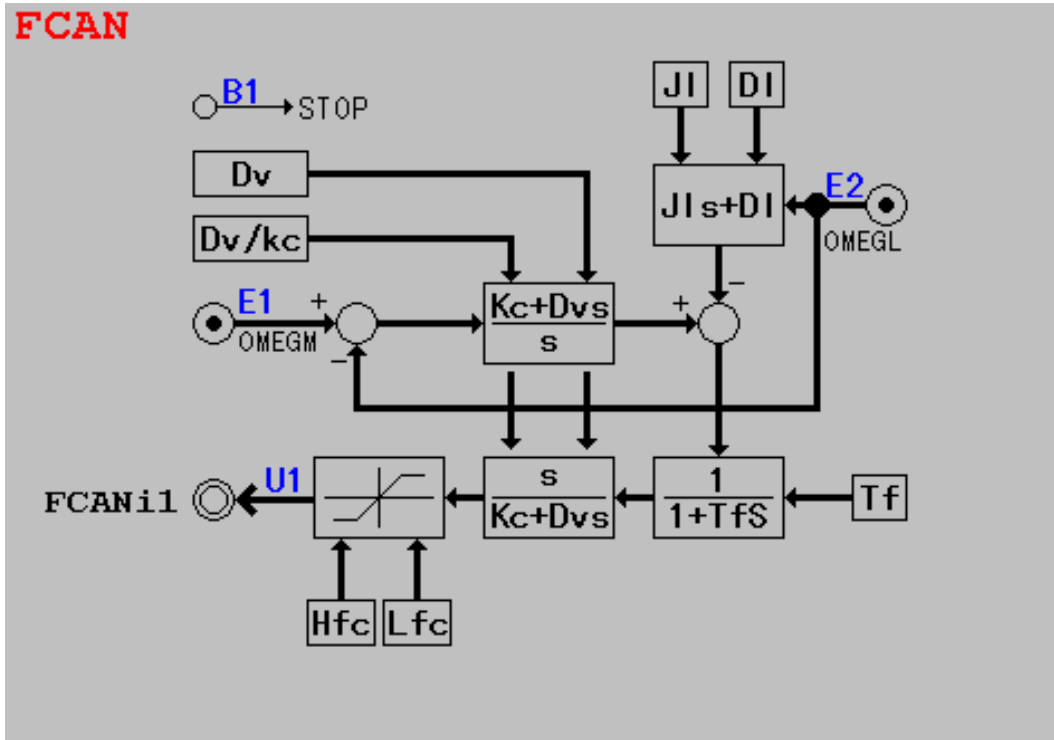
（2）ポート初期化用パラメータ詳細	
0000	汎用通信モジュール局番(ユニット番号、スロット番号)　ユニット1、スロット2の例)　102h
0001	ポート No(1:CH1 2:CH2 3:CH3)
0002 (未使用)･･･000C (未使用)	
000D	フレーム検出 0::なし　データが受信されれば受信完了となります。 1::可変長　先頭コードと終了コードで囲まれたデータを検出した時、受信完了となります。 2::固定長　受信データが受信バイト数に達した時、受信完了となります。
000E	受信バイト数固定長の時、受信バイト数を指定します。可変長の時は、”0”と指定します。
000F	先頭コードバイト数可変長の時、先頭コードバイト数を指定します。
0010	先頭コード1 可変長の時、先頭コードを指定します。
0011 先頭コード2、0012:先頭コード3、0013:先頭コード4、0014:先頭コード5	
0015	終了コードバイト数可変長の時、終了コードバイト数を指定します。
0016	終了コード1可変長の時、終了コードを指定します。
0017 終了コード2、0018:終了コード3、0019:終了コード4、001A:終了コード5	
001B:(未使用) ･･･001F:(未使用)	

種類	名称	シンボル	実行時間
データフロー言語 (関数 4) SHPC-115-Z のみ	モータ側キャンセレーション (現代制御)	MCAN	――――――
機能	モータのトルク指令と、実際のモータ回転速度からイナーシャ係数 Jn やダンピング係数 Dn を考慮し、求めた値との差を最小限にするよう制御します。		

入力信号

変数名	型	内容	設定範囲/単位	備考
B1	リレー	停止スイッチ		
B2	リレー	パワコン切替スイッチ		
E1	実数	トルク速度指令 Tref	%	
E2	実数	フィードフォワード Ffwd1	%	
E3	実数	フィードフォワード Ffwd2	%	
E4	実数	モータ角速度 Omegam	%	
E5	実数	FCAN 出力用 速度偏差 Dw	%	
E6	実数	パワコン係数 Zn	0.0～16.0	
Jn	実数	イナーシャ設定値	0.0～31.999	(注1)
Dn	実数	ダンピング設定値	0.0～0.999	(注2)
Tf	実数	フィルタ時定数	ms	デフォルト=10.0ms(注3)

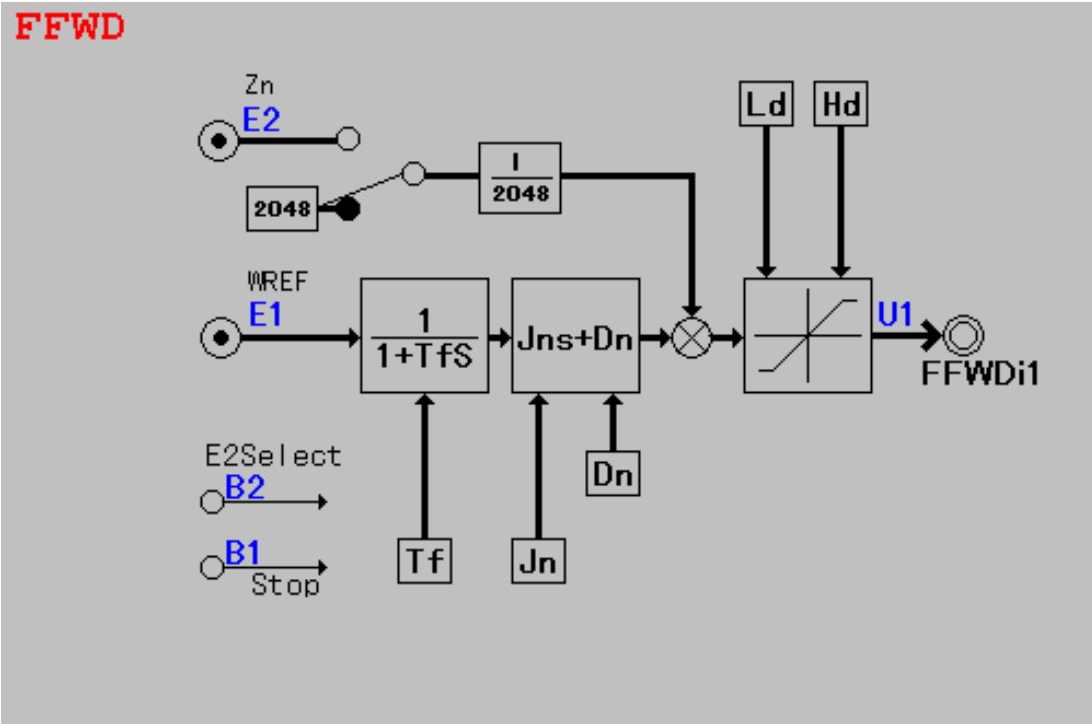
出力信号

変数名	型	内容	設定範囲/単位	備考
U1	実数	MCAN 出力1	%	
U2	実数	MCAN 出力2	%	

(注1)イナーシャ設定値(Jn)＝モータ軸換算のイナーシャ[kgm^2]×定格速度[rad/s]／定格トルク[Nm]
(注2)ダンピング設定値(Dn)＝モータ軸換算のダンピング[Nm･s/rad] ×定格速度[rad/s]／定格トルク[Nm]
(注3)Tf が、実行(演算)周期の 2 倍より短い値を設定した場合、正しい演算結果になりません。Tf が、サンプル時間の 2 倍より長い値になるように設定してください。

種類	名称	シンボル	実行時間
データフロー言語 （関数 4） SHPC-115-Z のみ	フレキシブル側 キャンセレーショ ン（現代制御）	FCAN ｆ	――――――
機能	モータのトルク指令、実際のモータ回転速度と、負荷側の回転速度から、負荷側のイナーシャ係数 Jln やダンピング係数 Dln、ねじり軸の各種設定値を考慮し、求めた値との差を最小限にするよう制御します。		

入力信号

変数名	型	内容	設定範囲/単位	備考
B1	リレー	停止スイッチ		
E1	実数	モータ角速度　Omegam	—	
E2	実数	負荷角速度　Omegal	—	
Dv	実数	ねじり軸ダンピング設定値	Dv：0.13～300.0	デフォルト=1.00（注1）
Dv/Kc	実数	ねじり軸遅れ時定数	1～1000ms	デフォルト=10ms（注1）
Jl	実数	負荷イナーシャ設定値	0.0～0.999	デフォルト=1.0（注2）
Dl	実数	負荷ダンピング設定値	0.0～0.999	デフォルト=0.0（注3）
Kf(=1/Tf)	実数	Tf：フィルタ時定数	Tf：1～1000[ms]	デフォルト=10.0ms（注4）
Lfc	実数	FCAN 出力下限値	%	
Hfc	実数	FCAN 出力上限値	%	

（注1）ねじり軸ダンピング設定値　Dv=ねじり軸のダンピング[Nm･s/rad]×定格速度[rad/s]/定格トルク[Nm]
　　　　ねじり軸バネ定数設定値　Kc=ねじり軸のバネ定数[Nm/rad]×定格速度[rad/s]/定格トルク[Nm]
（注2）負荷イナーシャ設定値（Jln）＝負荷のイナーシャ[kgm^2]×定格速度[rad/s]／定格トルク[Nm]
（注3）負荷ダンピング設定値（Dln）＝負荷のダンピング[Nm･s/rad]　×定格速度[rad/s]／定格トルク[Nm]
（注4）Kf の逆数である Tf が、実行（演算）周期の 2 倍より短い値を設定した場合、正しい演算結果に
　　　なりません。Kf の逆数である Tf が、サンプル時間の 2 倍より長い値になるように設定してください。

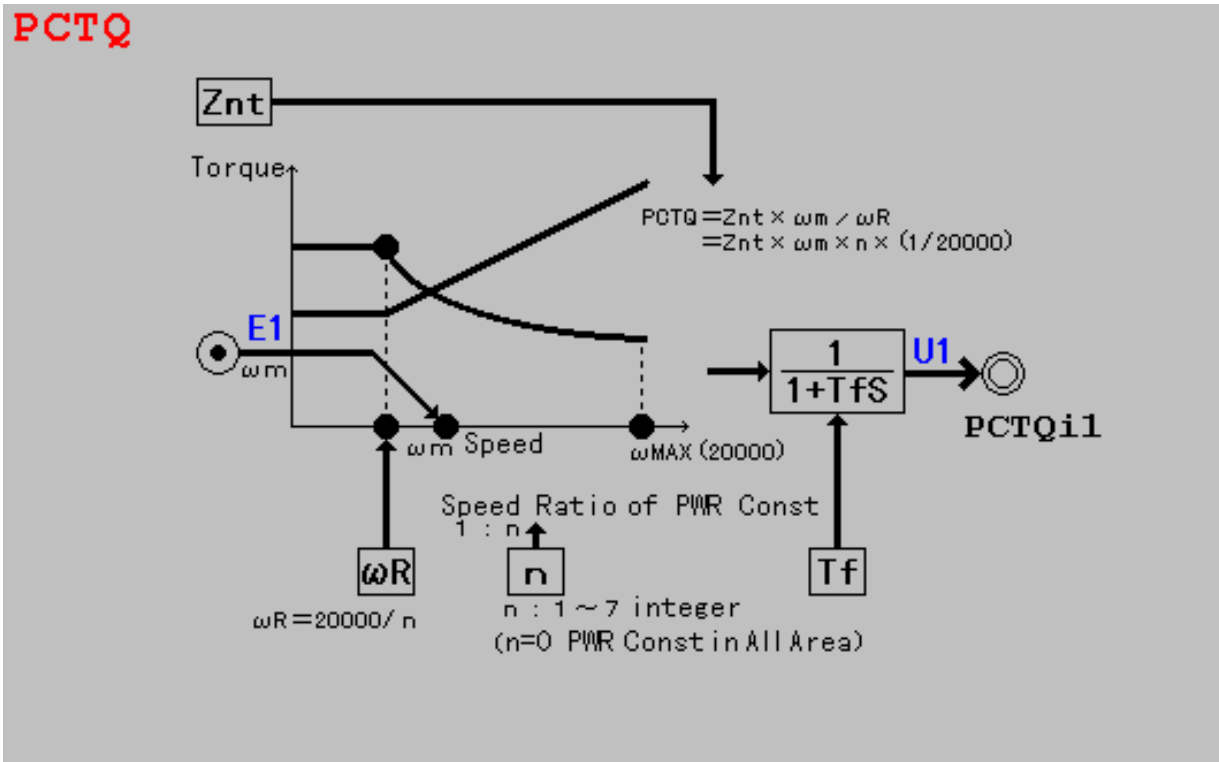
種類	名称	シンボル	実行時間
データフロー言語 (関数 4) SHPC-115-Z のみ	フィードフォワード (現代制御)	FFWD f	――――――
機能	モータ回転速度指令値に対して実際の回転速度の遅れ(差)を、イナーシャ係数 Jn やダンピング係数 Dn を考慮し補償します。		

入力信号

変数名	型	内容	設定範囲/単位	備考
B1	リレー	停止スイッチ		
B2	リレー	パワコン切替スイッチ		
E1	実数	モータ速度指令 Wref	%	
E2	実数	パワコン係数 Zn	%	
Jn	実数	イナーシャ設定値	0.000～31.999	デフォルト=1.0(注1)
Dn	実数	ダンピング設定値	0.0～0.999	デフォルト=0.0(注2)
Tf	実数	フィルタ時定数	(Tf:1～1000[ms]	(注3)
Hd	実数	出力上限値	-163.0%～163.0%	
Ld	実数	出力下限値	-163.0%～163.0%	

出力信号

変数名	型	内容	設定範囲/単位	備考
U1	実数	FFWD 出力	%	

(注1)イナーシャ設定値(Jn)＝モータ軸換算のイナーシャ[kgm^2]×定格速度[rad/s]／定格トルク[Nm]
(注2)ダンピング設定値(Dn)＝モータ軸換算のダンピング[Nm･s/rad] ×定格速度[rad/s]／定格トルク[Nm]
(注3)Kf の逆数である Tf が、実行(演算)周期の 2 倍より短い値を設定した場合、正しい演算結果になりません。Kf の逆数である Tf が、サンプル時間の 2 倍より長い値になるように設定してください。

種類	名称	シンボル	実行時間
データフロー言語 (関数 4) SHPC-115-Z のみ	パワコン係数 (トルク発生係数 の逆数)ブロック	PCTQ f	―――――
機能	トルク指令と実際のモータ回転速度を入力することで、基底回転速度以上の回転速度で電力一定となるように変換したトルク指令を出力します。		

入力信号

変数名	型	内容	設定範囲/単位	備考
E1	実数	入力 Wm	%	
wR	実数	定格(基底)速度	100.0～40000.0	
Znt	実数	ゲイン(トルク発生係数の逆数)	0.001～15.999	
Tf	実数	遅れゲイン(Tf:遅れ時定数)	1～10000[ms]	(注1)

出力信号

変数名	型	内容	設定範囲/単位	備考
U1	実数	PCTQ 出力	―	

(注1)Kf の逆数である Tf が、実行(演算)周期の 2 倍より短い値を設定した場合、正しい演算結果になりません。Kf の逆数である Tf が、サンプル時間の 2 倍より長い値になるように設定してください。

第6章　付　録

(付録1) シンボルと各名称

(1)LD 言語

表 6.1

A 接点	B 接点	論理反転	コイル	結合子ロード	結合子ストア
─┤├─	─┤/├─	─[~]─	─(　　)┤	①>─	─>①
ラベル	ジャンプ	リターン			
XXXXXX [L]─	─(JPXXXX)┤	─(RETURN)┤			

(2)データフロー言語(基本)

表 6.2

ロード	ストア＆ロード	ストア	a 接点	b 接点	c 接点
c 接点	コンペアハイ	コンペアロウ	コンペアイコール	上位優先	下位優先
論理積	論理和	論理排他和	加算	減算	乗算
除算	剰余	局所定数:整数	局所定数:実数		
		i	r		

(3)データフロー言語(関数 1)

表6．3

符号変換	1‘補数	絶対値変換	インクリメント	デクリメント	2 分の 1
-1	∿	✳	++	--	½
2 倍	2 乗	指数	平方根	ビットカウント	グレイコード バイナリー
×2	↑2	↑N	√	B.C	G.B

(4)データフロー言語(関数 2)

表64

不感帯	パターン	微分補償	位相補償	PI 補償	ARC
			θ		
S-ARC	算術平均	フィルタ	PID 補償	一時遅れ	ディレー
	x̄				
定周期パルス	変数設定 パターン	上下限リミッタ	ヒステリシス	無条件 サブルーチン	条件付 サブルーチン
				XXXXXX Sb	XXXXXX SB

(5)データフロー言語(関数 3)

表6.5

正弦	余弦	正接	逆正弦	逆余弦	逆正接
SIN f	COS f	TAN f	ASIN f	ACOS f	ATAN f
オンタイマ	オフタイマ	オン微分	オフ微分	バックラッシュ	バックラッシュ 補正
TSTD f	TRTC f	USUC f	DSDC f	BKLS f	BKLC f
スケーリング	バイナリーグレイ 変換	割り余り	整数変換	実数変換	
SCAL f	BTOG f	DIVMOD f	TODINT f	TOREAL f	

(6)データフロー言語(関数 4)

表6.6

チャンネルオープン	メッセージ送信	メッセージ受信	マトリクス	セット	リセット
M_OPEN f	M_SEND f	M_RECV f	MATRIX f	SET F	RESET F
データ転送	データ転送	カウンタ	汎用通信		
MOVW F	MOVWD F	UPDOWN F	C_FREE F		
モータ側キャンセレーション	フレキシブル側キャンセレーション	フィードフォワード	パワコン係数		
MCAN f	FCAN f	FFWD f	PCTQ f		

東洋電機製造株式会社

http://www.toyodenki.co.jp/

本　社	東京都中央区京橋二丁目 9-2（第一ぬ利彦ビル） 産業事業部　TEL.03(3535)0652～5　FAX.03(3535)0660,0664	〒104-0031
大阪支社	大阪市北区角田町 1-1（東阪急ビル） TEL.06(6313)1301　FAX.06(6313)0165	〒530-0017
名古屋支社	名古屋市中村区名駅三丁目 14-16（東洋ビル） TEL.052(541)1141　FAX.052(586)4457	〒450-0002
北海道支店	札幌市中央区大通西 5-8（昭和ビル） TEL.011(271)1771　FAX.011(271)2197	〒060-0042
九州支店	福岡市博多区博多駅南一丁目 3-1（日本生命博多南ビル） TEL.092(472)0765　FAX.092(473)9105	〒812-0016
横浜営業所	横浜市神奈川区鶴屋町二丁目 13-8（第一建設ビル別館） TEL.045(313)4030　FAX.045(313)4041	〒221-0835
広島営業所	広島市中区宝町一丁目 15（宝町ビル） TEL.082(249)7250　FAX.082(249)7188	〒730-0044
沖縄営業所	沖縄県中頭郡嘉手納町字屋良 1022 TEL.FAX.098(956)7314	〒904-0202

サービス網

東洋産業株式会社

http://www.toyosangyou.co.jp/

本　社	東京都千代田区東神田 1 丁目 10-6(幸保第二ビル) TEL 03(3862)9371　FAX 03(3866)6383	〒101-0031
大阪支店	大阪市淀川区西中島 4 丁目 7-4(新大阪生原ビル) TEL 06(6307)8181　FAX 06(6307)8185	〒532-0011
横浜支店	横浜市神奈川区鶴屋町 2 丁目 13-8(第一建設ビル別館) TEL 045(324)2356　FAX 045(324)3731	〒221-0835
名古屋営業所	名古屋市中村区名駅3丁目 14-16(東洋ビル) TEL 052(541)1150　FAX 052(586)4457	〒450-0002
九州駐在	福岡市博多区博多駅南 1 丁目 3-1（日本生命博多南ビル） TEL 092(413)6951　FAX 092(473)9105	〒812-0016
北海道営業所	札幌市中央区大通西5-8(昭和ビル) TEL 011(251)5611　FAX 011(271)2197	〒060-0042

本資料の記載内容は、予告なく変更することがあります。ご了承下さい。

2011-02 発行
QG18273C